LOW-LATENCY AND ROBUST
PEER-TO-PEER VIDEO STREAMING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jeonghun Noh
September 2010
© 2011 by JeongHun Noh. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This dissertation is online at: http://purl.stanford.edu/sp047bn0313
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Bernd Girod, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Fouad Tobagi

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Pierpaolo Baccichet

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

Peer-to-peer (P2P) systems have emerged as a promising and cost-effective transport solution for streaming video to a group of users in the Internet. In the P2P architecture, users not only consume video, but also forward it to other users. Thus, P2P systems scale better than client-server systems as users bring resources to the system. The challenge is to achieve low-latency and robust video dissemination by overcoming a number of adversarial aspects and challenges – peer dynamics, heterogeneous uplink bandwidth of peers, heterogeneous hardware and capabilities of peers, and peer-wise connection restrictions due to NATs/firewalls.

This dissertation presents Stanford Peer-to-Peer Multicast (SPPM), a P2P video streaming system. SPPM is designed to achieve low-latency and robust streaming by constructing an overlay of multiple complementary trees and dynamically rearranging the position of peers by Active Overlay Management in a distributed fashion. We also propose a distributed uplink bandwidth allocation scheme to alleviate peer-wise connection restrictions.

Next, we extend SPPM for providing playback control to users by time-shifted streaming. To perform time-shifted streaming, peers store past portions of video and forward them to other users when requested, thereby reducing server load. To further alleviate server load, we propose fast prefetching, by which peers can disseminate content quickly. Fast prefetching is shown to lower the number of requests to the server and to reduce video disruptions in case of peer failure.

Finally, we present a way to accommodate heterogeneous users, in particular, mobile users. To stream video to mobile users, video transcoding is often required to adapt video for the mobile users. We propose interleaved distributed transcoding.
(IDT), which allows a video stream to be transcoded at multiple peers that are more capable than mobile users. Transcoded substreams received at a mobile user are then assembled into a single video stream, which can be decompressed by any decoder that conforms to the H.264/AVC standard. IDT is shown not only to reduce computation required at a peer but also to achieve higher error resilience in case of peer failure or packet loss.
Acknowledgments

My graduate life at Stanford University has been so colorful, most of all, owing to the people I have met here.

My primary advisor, Professor Bernd Girod, has been a good role model. While working with him in the Image, Video, and Multimedia Systems (IVMS) group, he has helped me develop my skills related to research, technical writing, presentation, and interactions with other researchers. Professor Fouad Tobagi, my associate advisor, guided me to have a clear view of the identified issues and how I would address them in this research. Dr. Pierpaolo Baccichet, who is not only one of my reading committee members, but also a collaborator and close friend, has been a good source to learn about software engineering, and Italian culture.

Kelly Yilmaz was always available for assistance in administrative matters. Pat Burke took care of all of my requests for networking and computer issues associated with the IVMS cluster I administered. I also want to remember Evelin Sullivan for teaching me subtleties of English. The Korea Foundation for Advanced Studies supported me financially for the early years at Stanford.

I have been fortunate in working with many collaborators outside Stanford, including Sachin Deshpande, Ashish Khisti, Sachin Agarwal, Peter Pogrzeba, Kyu-Han Kim, and Jatinder Pal Singh.

My life at Packard Electrical Engineering building was full of interactions and discussions with the IVMS members. Eric Setton, whom I worked with for my first year at IVMS, taught me how to drive oneself constantly during the PhD program. Aditya Mavlankar spent a great amount of time to work together on our co-authored papers. Xiaoqing Zhu was the one I could always have an in-depth discussion on
network-related topics with. Although Chuo-Ling Chang and I had little overlap in
our research interests, we chatted a lot about our life after Stanford. Yao-Chung
Lin constantly encouraged me whenever my research showed a slow progress, saying
“don’t worry. You’ll be fine in the end.” Mina Makar and I worked together on the
mobile P2P streaming project and he was always a good reference when I needed
video-related knowledge. I would like to thank David Varodayan for being always
available for discussions.

My other friends took also a major part of my life at Stanford. Being with
Kwangmoo Koh, Hyung-June Lee, Jongwook Joo, and William Wu was a source of
joy and relaxation. My friends from KBSK, KCBC, BCSV, and NCBC should not be
neglected. I am thankful for their consistent love and support for my success. I was
blessed by God through them. My friends at Korea were all supportive and shared
my feeling during my ups and downs.

Last, but not least, my extended family provided me with physical and mental
support. My father and mother have fully sacrificed their lives for their children and
grandchild. My parent-in-laws had a consistent trust in me. Yuna, my daughter,
gives me joy that nothing is comparable to. Hee Won is not only my lovely wife, but
also my best friend who gives me wise and timely advice when I need it. I dedicate
this dissertation to them.
Contents

Abstract iv

Acknowledgments vi

1 Introduction 1

2 Background 6
   2.1 The Internet ................................................. 6
   2.2 Video Compression and Network Transmission ..................... 8
      2.2.1 H.264/AVC Video Coding Standard ...................... 8
      2.2.2 Video Adaptation ....................................... 9
      2.2.3 Video Transmission over the Internet ................... 9
   2.3 Client-server Systems for Video Streaming ..................... 11
      2.3.1 Content Delivery Network ................................ 11
      2.3.2 Server-based Video-on-Demand .......................... 12
      2.3.3 Server-based Mobile Streaming ......................... 13
   2.4 Peer-to-Peer Systems for Video Streaming .................... 13
      2.4.1 P2P File Sharing and DHT ................................ 14
      2.4.2 P2P Streaming ........................................... 15
      2.4.3 P2P Video-on-Demand ................................. 19
      2.4.4 P2P Mobile Streaming ................................. 19

3 Live Video Multicast 21
   3.1 Preliminaries .............................................. 22
List of Tables

3.1 The percentage of peers connected to all trees. Each row is obtained with the same peer join sequence. (NPS: no peer swapping, PS: peer swapping) .......................................................... 33
3.2 Peer uplink bandwidth distribution based on the measurements. (The data are based on Fig. D.8 in Appendix D) ................................. 35
3.3 Selection probability of peer type. The left column shows the parent type and the top row shows the child type. $0 \leq \alpha, \beta, \gamma \leq 1$ and $\alpha + \beta \leq 1$. .......................................................... 51
3.4 Peer characteristics used for Cases 1 and 2. $N$ is the total number of peers. The peer type distribution: $P(E_A) = a$, $P(E_B) = b$, and $P(E_C) = c$. $d_X$: average out-degree of Type $X$. $O_R$: over-provisioning factor. .......................................................... 54
3.5 Self-sustainability comparison of the equal-allocation $P_E(S)$ and the numerical solution $P_N(S)$ of (3.15). .................................................. 56
3.6 Numerical solutions $(\alpha_N, \beta_N, \gamma_N)$ and the corresponding $P_N(S)$ for selected configurations. $P_U$: self-sustainability likelihood achieved by the universal solution $\alpha_U = 0.34, \beta_U = 0.13, \gamma_U = 1.00$. $N_P$: number of peers. $a$: fraction of Type $A$, $b$: fraction of Type $B$, $c$: fraction of Type $C$, $N_T$: number of trees. $O_R$: over-provisioning factor. ................. 58
4.1 Effects of fast prefetching (computed over the first 900 seconds). Video quality is represented by the average PSNR of peers. The average PSNR is computed at the end of the simulation by taking the average of the PSNR value of the video frames displayed at all peers. For missing frames, copy error concealment (display of the previously decoded frame) is used. Server load is the average number of direct children of the server. Hops and throughput are used as parent selection criteria.

5.1 The average fraction of time during which a particular number of parents of a mobile node is missing. The total number of parents is set to 4. $\lambda/\mu$ is the ratio of the average parent lifetime to the average recovery time.

5.2 The average fraction of time during which a particular number of parents is missing given a different total number of parents ($\alpha$ is set to 30).

5.3 Probabilities used in the Gilbert-Elliot model.

5.4 Average CPU time required for encoding a frame at one of parents.

C.1 Peer characteristics. $N$ is the total number of peers. The peer type distribution: $P(E_A) = a$, $P(E_B) = b$, and $P(E_C) = c$. $d_X$: average out-degree of Type $X$. $O_R$: over-provisioning factor.


D.2 Symbols used in (D.3)

D.3 Average uplink bandwidth contributed by a peer based on contribution scenario.
List of Figures

1.1 Top: Client-server architecture. Users establish one-to-one connections to the server. Middle: IP multicast. A video stream is duplicated at the routers as it is forwarded at the network layer. Bottom: Peer-to-peer architecture. Users relay video to other users. One-to-one connections are established between users. .............................................. 3

3.1 A peer-to-peer overlay consisting of two complementary distribution trees. The encoded video is packetized and split into disjoint sub-streams, one for each distribution tree. .............................................. 26

3.2 An illustration of system clogging. Class A peer can have up to two child peers. Class B peers cannot have any child peers. Left: A system with available bandwidth. Right: A system experiencing clogging. All the paths from the tree root are blocked by Class B peers. ............ 30

3.3 (a) Unfavorable join sequence. (b) Favorable join sequence. ........... 31

3.4 Probability of false clogging detection with a different number of probes. 5% of peers in the system have available bandwidth at the time of probing. .................................................... 32

3.5 The left y-axis shows the clogging probability. The right y-axis shows the fraction of peers connected to the system when the un-leeching protocol is not employed. All peers connected to the system when the un-leeching protocol was used. ................................. 34
3.6 Probability of system clogging. Peers with their uplink bandwidths following the distribution in Table 3.2 arrive in the system. The over-provisioning ratio was 1.05. The clogging probability decreases as the server provides more bandwidth.

3.7 Probability of system clogging. Peers with their uplink bandwidths based on the distribution in Table 3.2 arrive in the system. The server accepts 4 direct children. The clogging probability decreases as the video bitrate decreases.

3.8 An illustration of child promotion. Left: Initial configuration. Right: Peer X switches its position with Peer Y. Peer Y becomes the new parent of Peer X. Peer Z, previously a child of Peer Y, connects to Peer X.

3.9 Left: a single tree. S denotes the source node. \( P_i, i = 1, 2, 3, 4, \) denote regular nodes. Right: \( N = 4 \) trees are built in the system, where \( N \) is the number of nodes except the source. The four nodes appear as the only intermediate node in one of the trees. \( S_i, i = 1, 2, 3, 4, \) denote the logical sources that correspond to the same physical source. The trees rooted at each logical source are logically isolated from each other.

3.10 An example of an out-degree-4 tree (\( N_t = 4 \)). \( P_i \) is a peer at depth \( i \) in the tree.

3.11 Worst end-to-end delay under different number of peers. The number of trees is equal to the out-degree except for degree \( N \), for which the number of trees is equal to the number of peers. \( (L = 1500 \text{ Bytes}, R_S = R = 1 \text{ Mbps}, T_p = 30 \text{ ms}) \)

3.12 Worst end-to-end delay under different number of trees. The group size is the number of peers in the system. The number of trees is equal to the out-degree. \( (L = 1500 \text{ Bytes}, R_S = R = 1 \text{ Mbps}, T_p = 30 \text{ ms}) \)

3.13 Mean worst end-to-end delay for the system with four trees. Peers start to experience larger delays at Point A and B as peers continue to join the system.
3.14 Mean worst end-to-end delays for different number of trees and peers.
The number of trees is equal to the out-degree. 

3.15 Mean worst end-to-end delays of trees for different number of trees and peers. Peer average lifetime is 120s. 50 simulations of length 900 seconds are run for each point in the curves.

3.16 Average tree height in terms of the logical hops from the tree root to all the peers. Peer average lifetime is 120s. 50 simulations of length 900 seconds are run for each point in the curves.

3.17 NAT/firewall type of peers

3.18 Access type of peers

3.19 The likelihood of self-sustainability over ($\alpha$, $\beta$) and ($\beta, \gamma$). Case 1: (a) and (b). Case 2: (c) and (d).

3.20 Self-sustainability of a system of 100 peers. Self-sustainability increases as the number of trees or the over-provisioning increases.

4.1 Video trajectories of live video, Peer 1, and Peer 2. Peer 1, an LS peer, requests the live video. Peer 2, a TS peer, requests the video delayed by $t_2 - x_2$.

4.2 A streaming window for a TS child that requests video from position $x_1$ at time $t_1$. $W_L$ is its lower end and $W_U$ is its upper end. $B$ is the pre-roll delay of play-out at the child. Packets whose time stamps fit in the streaming window are transmitted unless they have been sent already.

4.3 Initial video positions and request times for LS peers (Left) and TS peers (Right) that possess video at position $x$ at time $t$. Left: An LS peer whose arrival time is earlier than $x$ has video at position $x$ at time $t$. Right: A TS peer whose request time and initial video position are within the area surrounded by $(0, 0), (t - x, 0), (t, x)$, and $(x, x)$ on the time-video plot has video at position $x$ at time $t$. 
4.4 Video availability on the time-video plot. The value at \((t, x)\), predicted by the model, represents the expected number of peers available for video at position \(x\) at time \(t\) (\(\lambda = 1.125\) joins per second, \(1/\mu = 120\) seconds, and \(\alpha = 0.5\)). .................................................. 68

4.5 Model: video availability at 300 seconds. (\(\lambda = 1.125\) joins per second, \(1/\mu = 120\) seconds, and \(\alpha = 0.5\)) .................................................. 69

4.6 Model: video availability at 900 seconds. (\(\lambda = 1.125\) joins per second, \(1/\mu = 120\) seconds, and \(\alpha = 0.5\)) .................................................. 69

4.7 An example of fast prefetching for TS peers. The parent peer can serve two children concurrently. At time \(t\), the excess download at Child 1 due to fast prefetching is \(C_1(t) - C_1(t)\). .................................................. 70

4.8 Expected download at a child peer after time \(D\) when a child peer joining at time \(t\) connects to parent candidate \(i\). Left: child buffer lags behind parent buffer at time \(t + D\). Right: child buffer catches up with parent buffer. (\(q_i\): download rate of Parent Candidate \(i\) from its parent, \(l_i\): buffered video in bits at Parent Candidate \(i\), \(r_i\): upload rate from Parent Candidate \(i\) to the child peer.) ................................. 72

4.9 Uplink bandwidth partitioning. In Step 1, LS children are assigned the tree bitrate. In Step 2, TS children that have completed fast prefetching are assigned the tree bitrate. In Step 3, TS children that can perform fast prefetching are assigned the remaining uplink bandwidth evenly. Every peer except the server regularly performs bandwidth partitioning. .................................................. 73

4.10 Video availability averaged over 100 simulation runs. (\(\lambda = 1.125\) joins per second, \(1/\mu = 120\) seconds, and \(\alpha = 0.5\) are used.) ................................. 74

4.11 Video availability at 900 seconds. Averaged over 100 simulation runs. Only TS peers (\(\alpha = 0\)) joined the system at the rate of \(\lambda = 1.125\) joins per second. \(1/\mu = 120\) seconds. .................................................. 75

4.12 Comparison of server load over time (averaged over 10 simulation runs). ................................. 76

4.13 Normalized average download rate with respect to \(D\), look-ahead time. ................................. 78
5.1 A cascaded transcoder. An original video stream flows into the decoding unit. The intermediate processing unit transforms the decoded stream by performing frame rate conversion, downsampling, cropping, or any other preprocessing before encoding. The output of the encoder is the transcoded bitstream.

5.2 Interleaved distributed transcoding: an example of 4 parents (IDT encoders) with a GOP of 12 frames. The original video stream is divided into 4 substreams. The first frame of a GOP is encoded as an I frame by all parents. To avoid duplicate transmission, only Parent 1 transmits I frames. The I frames that are not transmitted are depicted with dotted boundaries. The solid arrows represent coding dependency. The dotted arrows depict the frame copy operation. The original stream is downsampled before encoding.

5.3 Decoding at a mobile node. In this example, one of 4 substreams, Substream 3, is assume to be missing due to parent loss. The frame copy bits from Substream 2 are used in constructing the assembled bitstream.

5.4 State-transition-rate diagram for a mobile node with \( K \) parents. \( 1/\lambda \) is the average recovery time. \( 1/\mu \) is the average parent lifetime.

5.5 Rate-distortion curves with and without packet loss for conventional transcoding (1 parent) and interleaved distributed transcoding (2 and 4 parents). (a) Foreman sequence; (b) Mother & Daughter sequence; (c) Container sequence; (d) Mobile sequence; (e) News sequence; (f) Paris sequence.

5.6 Video quality comparison of IDT and MDC. (a) Foreman sequence; (b) Mother & Daughter sequence; (c) Container sequence; (d) Mobile sequence; (e) News sequence; (f) Paris sequence.

A.1 The un-leeching procedure. Peer \( C \) requests peer swapping with Peer \( B \) (leech) to Peer \( A \) (leech’s parent).
A.2 The child promotion procedure. Peer X requests swapping with Peer Y. Peer Z attaches to Peer X when the swap is complete. 102
A.3 SPPM extension for mobile support. 103
A.4 System architecture of the SPPM mobile peer. 105

B.1 A random walk model for total available uplink bandwidth in the overlay. $B_0$ is the number of peers that the source peer can directly serve. Left: Example of good path. Right: Example of bad path. The reflected segment starting from $-B_0$ is depicted with a dotted line. 109
B.2 Probability of system clogging under incremental overlay construction. Source uplink bandwidth is the number of child peers the source peer can directly serve. 111

C.1 Distribution of $O_C$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_C} = 56.4, \sigma_{O_C} = 6.14$). No visible differences were found for various $\beta$ and $\gamma$ values. 116
C.2 Distribution of $O_B$ for $\alpha = 0.33, \beta = 0.15, \gamma = 1$. $P(S) = 0.903$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_B} = 135.65, \sigma_{O_B} = 4.67$). 116
C.3 Distribution of $O_B$ for $\alpha = 0.33, \beta = 0.41, \gamma = 0.61$. $P(S) = 0.765$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_B} = 135.62, \sigma_{O_B} = 8.27$). 116
C.4 Distribution of $O_B$ for $\alpha = 0.33, \beta = 0.67, \gamma = 0.22$. $P(S) = 0.880$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_B} = 138.77, \sigma_{O_B} = 7.52$). 116

D.1 An example system configuration: video is captured and encoded in real time. A session manager receives the encoded signal and delivers it to peers using the SPPM protocol. Peers consist of a set of super nodes and a group of users. 118
D.2 Number of concurrent users. Times are relative hours to the starting time. The actual starting time is specified next to Hour 0. Left: Day 1. Middle: Day 2. Right: Day 3

D.3 The probability mass functions (PMF) of peer lifetime for each day.

D.4 The PMFs of peer lifetime. Two distributions are fitted against data.

D.5 Conditional distribution of the departure of an active peer within the next 30 minutes.

D.6 Top: NAT type of peers. Bottom: Peer contribution for each type. (Symmetric: symmetric NAT. RESTR PORT: Restricted Port NAT. RESTR IP: Restricted IP NAT. FULL CONE: Full cone NAT)

D.7 User’s geographic distribution (all days). Users came from 93 countries.

D.8 Peer uplink bandwidth distribution.

D.9 Temporal evolution of the average uplink bandwidth. Left: Day 1, Middle: Day 2, Right: Day 3.
Chapter 1

Introduction

Over the last decades, advances in telecommunication networks and video coding have made video streaming increasingly popular [206, 197]. Video streaming allows a user to watch video in real time while the video is being transmitted to the user across the network. We consider the problem of streaming video to a group of users simultaneously in the Internet. IP multicast [176] is a network layer protocol used to efficiently route data from a source to multiple destinations. In case of video streaming, a video source sends video packets to a multicast address, and the routers forward the packets to the users who are interested in the video.

Although the IP multicasting protocol employs network resources efficiently, end-to-end “global” multicast service is not yet available. As an alternative, unicast routing has been widely used to deliver video in the Internet. Typically, media servers are used to stream video to users by using multiple unicast connections. In such a client-server architecture, users (as clients) merely consume video transmitted from the server. Deployment and operations of media servers are relatively costly, and the system capacity, measured by the number of concurrent users being served, is limited by server resources.

In contrast to the client-server architecture, the peer-to-peer (P2P) architecture allows users to contribute their resources to the system. In the P2P system, users (as peers) not only consume video, but also route it to other users. Thus, P2P systems scale well as users bring resources such as uplink bandwidth, storage space, and
computing power. The client-server architecture, IP multicast, and P2P architecture are illustrated in Fig. 1.1.

This thesis presents the Stanford Peer-to-Peer Multicast system (SPPM), a P2P system as a potentially cost-effective solution to streaming video to a group of users. Since peers are closely involved with video dissemination, some of the characteristics associated with peers influence the system design. For instance, some peers may not have sufficient bandwidth to forward the video at the full bitrate. This is largely due to the low uplink speed of hosts connected to the Internet via xDSL [91] or cable networks [52], where a high downlink speed is appropriate for downloading data from a server in the client-server architecture. Another unique aspect of peers is their dynamics in group membership – peers may join or leave the system anytime without prior notification, which renders P2P-based streaming challenging. Also, due to the limited number of IP addresses (IPv4) and security issues, a significant number of peers are behind Network Address Translators (NATs) [69] or firewalls [38], thereby not directly accessible from other peers. Moreover, the transmission delay from the video source to users tends to be longer than IP multicast since video is relayed from peer to peer to reach all users. The aim of the research presented here is to design a low-latency and robust P2P streaming system by overcoming these adversarial aspects and challenges.

Our contributions presented in this thesis are the following:

**P2P system that achieves low-latency and robust video streaming** We propose a P2P system that allows a video source to stream video to a group of users beyond its own uplink bandwidth. The system allows peers to incrementally construct an overlay consisting of multiple complementary trees [138, 40] in a decentralized fashion as peers join the system. We identify the problem of “clogging”, a system condition that no more bandwidth is available for new peers. We propose *Active Overlay Management*, which not only overcomes clogging but also reconfigures the overlay in order to achieve low transmission delay and high robustness under peer dynamics.
Figure 1.1: Top: Client-server architecture. Users establish one-to-one connections to the server. Middle: IP multicast. A video stream is duplicated at the routers as it is forwarded at the network layer. Bottom: Peer-to-peer architecture. Users relay video to other users. One-to-one connections are established between users.
NAT-aware bandwidth allocation algorithm. Due to the presence of NATs and/or firewalls, a connection between a certain pair of peers may be rendered impossible. A P2P system is said to be self-sustainable if all peers connect to the overlay and receive video. We first investigate the self-sustainability of a system for a given group of peers by considering their access type and uplink bandwidth. We then propose a distributed bandwidth allocation scheme that maximizes the likelihood of system self-sustainability.

An extension of the P2P protocol for user playback control. We extend the SPPM system to support user’s playback control. With playback control, users can individually pause a live video stream, rewind to an arbitrary position (even to contents prior to their time of joining), and fast forward to the latest position of the live video. To achieve this in a P2P fashion, peers store received video packets in their local buffer and forward them at a later time when requested by other peers, which is called time-shifted streaming. We also analyze the video availability, which refers to how many peers possess video contents of a particular position. Motivated by the analysis, we propose fast prefetching and a parent selection scheme suitable for fast prefetching in order to reduce server load by disseminating video quickly.

Distributed video transcoding scheme for mobile streaming. We address a problem of incorporating heterogenous users, in particular, mobile peers, in SPPM. In order to stream video to mobile users, transcoding is often required to render video suitable for their display size, computing power, and media capabilities. We propose interleaved distributed transcoding (IDT), a video encoding scheme that allows peers more capable than mobile users to perform transcoding collaboratively. We show that the proposed transcoding not only reduces computation at a peer but also achieves error resilience in case of peer failure or packet loss. We also analyze the effect of distributed transcoding in the presence of peer failure.

The remainder of the thesis is structured as follows: In the next chapter, we review the existing literature on video streaming and peer-to-peer systems. In Chapter 3,
we present the characteristics of peers that need to be taken into consideration for P2P live streaming systems. Then, we describe the SPPM system in detail, including its basics and active overlay management. We also discuss how to improve peer-wise connectivity in the presence of NAT devices and firewalls. In Chapter 4, we present an extension we made to SPPM in order to enable time-shifted streaming. We present an analysis of video availability, which shows how many peers possess video contents at a particular point in time. Next, we describe fast prefetching, a method to improve video availability. Fast prefetching allows peers to disseminate video faster, and the improved video availability is shown to lead to a lower server load. To facilitate video dissemination with fast prefetching, a new parent selection algorithm, called maximum throughput, is proposed and its effects are demonstrated with extensive simulations. Chapter 5 presents a way to support heterogeneous users, in particular, mobile users. Interleaved distributed transcoding, which allows multiple fixed nodes to perform transcoding for a mobile device, is presented. We analyze the effect of distributed transcoding under peer churn. Extensive simulations show that the performance of the proposed scheme under packet loss due to peer churn and adverse wireless channel conditions. Finally, Chapter 6 presents conclusions and suggestions for future work.
Chapter 2

Background

This chapter presents a survey of video streaming. In Section 2.1, we provide an introduction to the Internet. Some aspects of the Internet that affect peer-to-peer systems are also described. In Section 2.2, we provide an introduction to the H.264/AVC video compression standard and advances in video transmission over the network. Section 2.3 reviews advances in video streaming based on the client-server architecture. In Section 2.4, we present a survey of peer-to-peer systems and advances in video streaming based on the peer-to-peer architecture.

2.1 The Internet

The Internet is a global data network that consists of interconnected packet networks supporting communication among host computers using the standard Internet Protocol suite (TCP/IP). The protocol layers of the Internet are the link layer, Internet layer, transport layer, and application layer [37]. Each layer is associated with its supporting protocols and acts as an abstraction to the adjacent layer. Among the layers, the Internet layer uses the Internet Protocol (IP), which provides a best effort service for routing packets from source to destination. The transport layer sits on top of the network layer, and its two major protocols are the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) [103]. TCP provides an error-free channel that delivers data bytes in the order in which they were sent, which is suitable
for reliable data transmission. UDP, on the other hand, provides a best-effort delivery of a datagram with no guarantee. Since UDP allows an application layer to carry out its own flow control as well as media-specific error control, it is often adopted for delay-sensitive and loss-tolerable media transmission.

While the Internet Protocol routes an IP diagram to a single destination, IP multicasting routes an IP datagram to a group of hosts identified by a multicast IP address [61]. A multicast datagram is delivered to all members of its destination host group with the same “best-efforts” reliability as regular unicast IP datagrams. Many multicast routing protocols, such as DVMRP [182], MOSPF [131], and PIM [15], differ in the construction of multicast packet delivery trees at multicast routers. In its early stage, IP multicasting was tested and used on the Internet Multicast Backbone (MBONE). The MBONE is an overlay network where multicast-enabled hosts are connected to other multicast-enabled hosts by point-to-point connections called “tunnels.” Recently, IP multicast has become increasingly popular in video streaming services in a private network, such as IPTV [129, 102].

A host computer, or simply “host,” is a networked computer that consumes communication services in a network. A majority of hosts are connected to the Internet via residential broadband networks such as Digital Subscriber Lines (DSL) and cable Internet [63]. More than 283 million people use these networks worldwide [136]. In the United States alone, more than half of all Internet users connect to the Internet via residential broadband networks [134]. The data throughput of DSL services typically ranges from a few hundreds of Kbps to 20 Mbps in the direction to the subscriber (downlink), depending on DSL technology, line conditions, and service-level implementation. In ADSL, the uplink bandwidth (in the direction to the network) is much lower than the downlink bandwidth. Cable Internet services also provide asymmetric speeds for uplink and downlink. It is known that the reliability and the performance of Internet applications – including voice-over-IP (VoIP), online games, IPTV, and peer-to-peer content delivery systems – depend crucially on the characteristics of broadband access networks [111, 32].

In practice, hosts are often connected to the Internet via NATs or firewalls. Network Address Translators [73] allow multiple hosts on a private network to access
the Internet using a single public IP address. Since NATs alleviate the problem of
IPv4 address exhaustion, they are prevalent in home and corporate networks [38]. A
firewall [88] is a system that enforces an access control policy between two networks.
It blocks unauthorized access from malicious users while permitting authorized com-
munications. NATs and firewalls are suitable for the client-server communication in
which the client is on a private network and the server is in the public space. However,
NATs and firewalls break the originally envisioned model of IP end-to-end connectiv-
ity across the Internet, and introduce complications in communication between hosts
[111, 142, 19, 190].

2.2 Video Compression and Network Transmission

In this section, we introduce the H.264/AVC video compression standard and describe
some aspects of the standard, which are relevant to this thesis. We also review
advances in video adaptation and video transmission over the network.

2.2.1 H.264/AVC Video Coding Standard

H.264/AVC [92], the state-of-the-art video compression technology, is the block-
oriented motion-compensation-based codec standard developed by the ITU-T Video
Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts
Group (MPEG). The final draft of the first version of the standard was completed
in May 2003. H.264/AVC compresses video, a temporal sequence of digital images,
by exploiting spatial redundancy and temporal redundancy in the video sequence.
H.264/AVC can achieve similar video quality as H.263 while requiring about 50% of
the bit-rate [101]. Nevertheless, its basic design is largely based on earlier standards
including H.261, MPEG-1, H.262, H.263 and MPEG-4 [90, 89, 93, 181, 14]. Details
of the improvements made in H.264/AVC can be found in [92].
2.2.2 Video Adaptation

Video adaptation converts an original video bitstream to a new bitstream for a different encoding standard, smaller spatial resolution, reduced frame rate, or reduced quality (due to coarser quantization) [25, 174, 192, 17]. In [36], temporal transcoding, by which video frames are skipped and the motion vectors in the following frame are modified in order to reduce the bitrate, is conducted according to a channel condition. For rate control, rate-distortion based frame transcoder [120] and frame skipping transcoders with motion vector update [141, 108] have been proposed. A centralized transcoding algorithm is proposed in [99, 98] for multi-user video streaming over a wireless LAN. In [99, 98], the rate of transcoded video bitstream is dynamically adjusted by the central server in order to maximize the rate-distortion performance by considering the video characteristics and network conditions. Scalable video coding has its obvious merits when an encoded video signal needs to be transmitted to mobile users with different bitrate or reception capabilities and/or the encoding cannot be done or is not economically viable for each and every receiver [156, 188].

2.2.3 Video Transmission over the Internet

The quality of service limitation is still one of the major challenges for real-time video communication over the Internet. Since the IP protocol offers a best-effort service, packets may be delivered with arbitrary delay or may even be lost. Packet loss or late packet arrival may result in poor picture quality and frozen frames in video. This section provides a brief list of advances in error corrections, and network-aware video compression.

Error Control and Concealment

Feedback-based error control is a very effective approach to robust video transmission when a feedback channel is available and the extra delay incurred by such schemes is permissible. In this approach, a sender retransmits a lost or damaged packet when the sender receives a feedback message from the receiver [143, 151, 126]. In a more sophisticated framework known as RaDiO (Rate-Distortion Optimization) [47], proactive
packet scheduling is performed to determine when and which packet to send to maximize the expected video quality at the receiver. In a related approach named CoDiO (Congestion-Distortion Optimization) [161], self-congestion due to retransmission is taken into account for packet scheduling. Our prior work in [164] extended CoDiO to tree-based P2P video streaming.

If the delay required for the feedback-based schemes is prohibitive, forward error correction (FEC) may be performed, in which a portion of the total transmitted bitrate is dedicated to parity information which enables the decoder to correct transmission errors. The most popular channel code used to provide FEC is the systematic Reed-Solomon code [149], in which packets of parity symbols are appended to the video packets prior to transmission. FEC, however, may suffer from the rapid reduction in picture quality, called the “cliff effect,” when the number of symbol errors is too high. To alleviate the cliff effect, the authors of [148] introduce Systematic Lossy Error Protection (SLEP), which transmits an additional bit stream generated by Wyner-Ziv coding.

When some packets miss the playback deadline, the receiver may not be able to decode video frames. Then the receiver conceals the lost portion of the video based on several methods. Error-concealment minimizes the negative impact of late or missing frames as well as the visual quality degradation caused by the missing frame. Some error concealment schemes use estimation of coding modes and motion vectors [26, 169] and spatial or spatio-temporal interpolation [20, 33].

Network-aware Video Compression

While the latest video coding standards have tools for robust streaming such as video slices and data partitioning, several video coding schemes specifically aim to gain error resilience over a lossy network, or to adapt to network conditions or heterogeneous clients during transmission. For instance, a video signal may be encoded into a number of layers that can be incrementally combined to provide progressive refinement. This technique is called layered coding. Dropping layers in the network (e.g., selectively forwarding only the number of layers that any given link can manage) adapts video to varying network conditions without the source’s intervention [76, 128].
CHAPTER 2. BACKGROUND

Multiple Description Coding (MDC) exploits the availability of multiple paths between the source and the receiver [78, 24, 23]. In MDC, a signal is encoded into multiple descriptions that are independently decodable, which may lead to better quality if more descriptions are available in decoding. The performance of MDC is examined in [23] when applied to CDN, or in the context of P2P [194]. Layered coding and MDC are used together to achieve better system performance in [48, 44, 171]. The authors of [119] survey adaptive techniques used in video multicast.

SVC [157, 158] is an extension to the H.264/AVC, which allows for video services with lower temporal and/or spatial resolutions, and/or reduced fidelity are rendered possible by the transmission of partial bit streams. Hence, SVC provides functionalities such as graceful degradation in lossy transmission environments as well as bitrate, format, and power adaptation. To efficiently handle heterogeneous downlink users, the authors of [28] study the performance of SVC combined with P2P streaming.

Some researchers developed video coding algorithms tailored for efficient communication in peer-to-peer environments. For instance, the authors of [121] present an efficient substream encoding algorithm with P2P VoD in mind.

2.3 Client-server Systems for Video Streaming

Video streaming allows a user to watch video in real time while the video is being transmitted to the user across the network. The traditional approach to supporting streaming media and Web traffic alike is to use the client-server architecture, where a single server is in charge of serving clients.

2.3.1 Content Delivery Network

Content Delivery Networks (CDNs) [140] are a large-scale content distribution network. In a typical CDN, the content delivery infrastructure consists of a set of edge servers that deliver copies of content to users. The distribution infrastructure copies content from the origin server to the cache servers or edge servers. The request-routing infrastructure is responsible for directing client requests to appropriate edge servers.
It also interacts with the distribution infrastructure to keep an up-to-date view of the content stored in the CDN caches/replicas. Once directed to a nearby edge server, a user establishes a unicast connection to the server. Examples of commercial CDNs include [1, 8, 3, 2].

In [145, 51, 94, 147], placing the replicated content in the CDN networks is studied. If the content is a live media, real-time adaptation is required for transmitting the content to the replication servers in the CDN [30, 117, 167, 198]. The server may be serving a large number of concurrent users, but the communication is still one-to-one from a host’s perspective. Typically, one media server can serve from a few dozens up to hundreds depending on the bitrates of video contents and the server’s network capacity. Usually deployment and operations of media servers are costly, and the system capacity, measured by the number of concurrent users to service, is limited by the server resources.

2.3.2 Server-based Video-on-Demand

Video-on-Demand (VoD) systems allow users to select and watch video content on demand. With playback control, also known as trick mode, users can also pause a video stream, rewind to an arbitrary position, and fast forward. Unlike live video, video files requested on-demand are pre-encoded and stored at video servers. If the available uplink bandwidth at the sender exceeds the video bitrate, the user can receive video faster than the play-back speed [166]. Traditionally VoD services were provided by servers. For instance, Chaining [168] and Patching [85] employ media servers and IP multicast, where IP multicast is used to reduce server workload. The authors of [155] study trade-off between the system profit and user experiences in a near-VoD system. In [146], patching and batching of server-based VoD service are augmented by taking into consideration heterogeneity in receivers.

2.3.3 Server-based Mobile Streaming

In the last decade, mobile devices, such as smart phones or Personal Digital Assistants (PDAs), have become ubiquitous in our daily lives. However, live streaming
to mobile devices is still a challenging task due to their heterogeneity. In typical streaming systems, video transcoding is performed at servers to adapt video to mobile users. In [67, 108], server-based adaptive video transcoding systems for mobile users are proposed. A video proxy, located at the edge of two or more networks, adaptively transcodes video, considering the network conditions and constraints of mobile users in the GPRS network. The authors of [187] also propose a server-based video transcoding scheme for mobile users. The client proxy collects mobile client profiles and requests transcoding to the video proxy. Transcoded videos are then sent back to the client proxy, and the client proxy relays them to each client. The authors of [68] compare solutions that extend IP multicast to support mobile users. Since IP multicast trees are constructed at the network layer, these solutions often focus on the reduction of the overhead associated with the reconfiguration of the trees due to the mobility of mobile users.

2.4 Peer-to-Peer Systems for Video Streaming

Peer-to-peer (P2P) computing deals with instability, transient populations, fault tolerance, and self-adaptation [21, 79]. Broadly speaking, a usenet newsgroup network is an example of P2P content distribution [70]. When a user posts to one news server, the message is stored locally. Then, the message is disseminated to the other news servers asynchronously. Internet Relay Chat (IRC) [100] is another example of P2P content distribution: it is a real-time Internet text messaging based on the client-server architecture. IRC supports many-to-many group communications by relaying user text messages via dedicated IRC servers. In contrast to the P2P systems consisting of servers, this dissertation considers P2P systems where a peer is a networked user device that can display streamed video, which encompasses computers, netbooks, set-top boxes, and mobile devices.

Since P2P systems rely on cooperation among users, the system performance is inevitably degraded when users are reluctant to cooperate due to concerns about security or lowered performance. Providing incentives for user cooperation is an interesting topic, but it is beyond the scope of this dissertation. Readers may refer
to [71, 123, 77, 118] to see how incentive mechanisms can discourage free riders in P2P systems. Regarding security, P2P systems share most of their security needs with common distributed systems: trust chains between peers, resource access from other peers, and encryption. A malicious node might give erroneous responses to a request, both at the application level or at the network level. More seriously, an attacker may mix polluted chunks into the video bitstream, called pollution attack, thereby degrading the quality of the rendered media at the affected peers. Readers are referred to [183, 114, 185, 130, 62] for more details.

2.4.1 P2P File Sharing and DHT

P2P systems have become tremendously popular since Napster [9], an online music P2P file sharing service, appeared in 1999. Napster was the first of several massively popular P2P file distribution systems. It was based on central servers to maintain lists of online peers and the files they provided. Actual file download was performed from the source peer to the destination peer. Since Napster’s success, decentralized peer-to-peer systems such as KaZaA [7] and eMule (upgraded eDonkey) [4] have been deployed.

In the eMule system, the file search relies on Kademlia [127], a Distributed Hash Table (DHT). DHT protocols, such as Chord [172], Pastry [154], and Tapestry [204], all address the problem of efficiently locating the node that stores a particular data item. The DHT protocols provide a look-up service: given a key, this service maps the key onto a node. DHTs then form an infrastructure that can be used to build more complex services, such as distributed file systems, peer-to-peer file sharing, and content distribution systems.

2.4.2 P2P Streaming

P2P streaming was initially proposed as a deployable alternative to IP multicast because it can be used in networks that have no support for IP multicast. P2P multicast reduces the overhead associated with setting up and maintaining a multicast tree at the routers. Also, it is much easier for the end users to support the provisioning,
maintenance, and tracking of Quality-of-Service (QoS) over a unicast connection than over an IP multicast tree [50]. P2P multicast leverages the peer-to-peer concept for distributing time-sensitive content such as video. Since a P2P multicast network consists of unicast connections (e.g., TCP or UDP) on top of an existing network, it is often called overlay multicast or application layer multicast. Since users bring resources to the system, P2P multicast lowers server load, thus scaling well. The performance of P2P systems for large-scale sessions, including IPTV applications and commercial systems, was studied in [170, 87, 80, 18], where the authors report that in most cases P2P systems have enough resources, stability, and adaptability in the presence of peer dynamics and network congestion. Recently, the standardization of peer-to-peer streaming is being performed by IETF [6]. P2P multicast, P2P multicast streaming, and P2P streaming are interchangeably used in this dissertation.

**Data-Driven Swarming Approach**

Data-driven, or gossip-based P2P streaming extends the technique used in BitTorrent [53] with the notion of delay and rate constraints. In a gossip-based protocol, every node periodically exchanges its data availability information with a set of partners. Then, each node independently selects its neighbors so as to form an unstructured overlay network. Gossip-based P2P systems include PRIME [124], Tribler [144], DoNet/CoolStreaming [203], ppLive [10], SopCast [11], and Chainsaw [139]. One of the challenges associated with the data-driven approach is the excessive start-up time, which may be up to a few minutes [111, 19]. The authors of [111, 83] studied gossip-based systems based on system logs or dedicated crawlers. They found that pure gossip-based approach may lead to a long start-up delay, which can be significantly reduced by the push-pull approach [111]. Similarly, the authors of [16] found that an overlay-driven push approach is more suitable for a low start-up delay. The overlay-driven push approach will be discussed in detail in the next section.
CHAPTER 2. BACKGROUND

Overlay-Driven Push Approach

Overlay-driven, or tree-based P2P streaming is largely based on explicitly establishing a data distribution overlay. Tree-based overlays have been a popular choice because a tree structure spans all peers, systematically avoiding the delivery of duplicate packets. In this approach, one or multiple spanning trees are constructed to connect all peers participating in a session. The video source is the root of each tree, and peers potentially offer to forward, fully or partially, the video traffic to other peers.

In the early systems, a single multicast tree was built for data distribution [74, 95, 107, 177, 57, 29]. M-RTP [54] established many one-to-one RTP connections to set up the multicast session over a set of unicast connections between the participants in the multicast session. ESM, designed for a small-scale teleconferencing session, builds a mesh overlay among peers first, and then builds an overlay spanning tree for each sender on top of the mesh overlay [50].

Scribe [39] and Bayeux [207] build a single overlay tree on top of a DHT-based index overlay, and SplitStream [40] builds multiple trees on top of a DHT-based index overlay. In SplitStream, the DHT-based overlay is built and maintained by Pastry [122]. However, compromises made at the lower levels of the stack may result in a weak platform for a different class of applications built on top of it because the underlying DHT layer was designed for a class of applications that require no strict time constraints nor bandwidth requirements [34].

Since a single tree is vulnerable to peer dynamics, the multiple tree approach has been investigated by many researchers. The authors of [133] discuss the benefit of multiple streaming by path diversity. In [116], the advantage of path diversity in conjunction with adaptive video encoding based on receiver feedback is studied. Most multiple-tree overlay systems build trees independently of any index overlay. CoopNet [138] takes a centralized approach for constructing either random or deterministic node-disjoint multiple trees and achieves robust streaming by encoding video using MDC. ChunkySpread builds multiple trees among peers [178]. Unlike CoopNet, it builds trees in a distributed manner, thereby avoiding a single point of failure at the central server. GridMedia, a push-pull system, builds multiple trees for pushing down video packets while relying on pulling (e.g., gossip-based algorithm) for reliable
transmission [199]. Although many P2P systems support one-to-many communication, some systems based on multiple-shared trees [175, 31] support many-to-many communication.

Hybrid Systems

In this section, we review some of the systems that combine two different types of systems we discussed earlier: the integration of CDN and P2P, and the integration of data-driven P2P and overlay-driven P2P. Due to their highly complementary advantages, the integration of CDN and P2P compensates for their individual weaknesses. For instance, [160] employs a small number of servers (CDN) to address disruption due to peer churn and long start-up delay, which are two major weaknesses of pure P2P networks. In [45], the authors find that in an IPTV environment, a P2P incentive model performs better than any other model when a large population of users gathers. On the other hand, a CDN model generally is economically more efficient when the number of users is low. Other P2P-CDN hybrid systems treat the CDN as a reliable data dissemination channel and as an indexing server (user membership management) while employing P2P as a way to reduce the load imposed on the CDN. Depending on the architecture, CDN is either used at the beginning of a video session, when the number of users is small, or CDN is complemented by P2P, or vice versa [193, 97, 195].

Another type of hybrid P2P system combines trees and gossip-based meshes [186, 125]. In particular, Zhang et al. integrate data-driven and overlay-driven strategies, called push-pull [199, 200, 201]. In their work, the gossip-based algorithm is used to construct the overlay. Since the sheer pulling approach based on the gossip algorithm cannot achieve low latency owing to redundant latency accumulated at each hop, a push-pull mechanism is proposed. In the push-pull mechanism, each node uses the pull method as a startup, and after that each node relays a packet to its neighbors as soon as the packet arrives, without any explicit request from the neighbors.
CHAPTER 2. BACKGROUND

Performance Study

Much effort has been made to understand the behavior of P2P streaming systems and to improve the system performance in terms of the resource utilization, reliability, and end-to-end transmission delay. Some theoretical studies on the fundamentals of a P2P streaming system have provided insights into the system behavior and performance [109]. Yi Cui et al. provide a set of analytical frameworks to improve the performance of a P2P streaming system. In [41], the authors incorporate a generalized flow approach to maximize the system throughput. They also strive to find the max-min rate allocation in the overlay, which is pareto-optimal in terms of network resource utilization and max-min fairness. The overlay then rearranges nodes in the tree to achieve the rate allocation among the nodes [57]. In [58], the authors investigate how to maximize the aggregate utility of all receivers, where a peer’s utility is defined as a function of its streaming rate. The basic assumption they applied to these studies was that peers benefit from higher downlink streaming rates. In [59], the authors derive a theoretical model that captures the end-to-end delay by taking into account the physical proximity among peers. Recently, the authors incorporated the effect of peer churn in their theoretical framework [55].

Several overlay rearrangement algorithms are evaluated with respect to reliability, delay, and protocol overhead in [191, 96]. In [132], the parent selection scheme based on actual measurements, such as bandwidth and round trip time, is proposed. Parent selection/switch based on uplink bandwidth and lifetime is discussed in the context of multi-tree overlays in [35]. The authors of [189] reduce the end-to-end transmission delay in conjunction with a rateless code, which eases coordination of multiple streams received from random peers. As in [58], [104] proposes a set of algorithms to maximize aggregate uplink bitrates. MutualCast [113] achieves the full utilization of peers’ heterogeneous uplink bandwidth. However, with MutualCast, peers allocate smaller uplink bandwidth per child peer as more peers join the system, thereby increasing end-to-end delay. The authors of [60, 72] study the end-to-end loss characteristics for live multicast P2P protocols based on multiple complementary trees.
2.4.3 P2P Video-on-Demand

As in server-based live streaming, servers deployed for VoD services may fail in the presence of a large population of users. Thus, many P2P-based VoD systems have been proposed to achieve scalable services. In order to achieve low workload on servers, the resources in the P2P network are leveraged for efficiently taking advantage of uplink bandwidth and storage contributed by peers [56, 184, 81, 64]. For VCR-like operation (temporal random access), either distribution overlay is combined with look-up overlay [56, 184], or a separate look-up overlay is built to avoid a central look-up service [46, 196, 82]. In [81, 65, 86], viewers are able to watch a pre-recorded video from the beginning of the content. Dynamic Skip List [184] and oStream [56] provide users random temporal access to video content. The mesh-based P2P VoD systems are studied in [194, 121, 22]. The authors of [86] exploit the extra uplink bandwidths and peer storages by downloading video faster than playback speed. In [205], a user-log based prefetching algorithm is proposed. PPB, Peer-to-Peer Batching, combines peer-to-peer and IP multicast by using IP multicast for batching while a peer-to-peer network is used to provide the initial portion of video to new peers [84].

2.4.4 P2P Mobile Streaming

Although server-based systems [67, 187] work reasonably for a moderate number of users, transcoding servers can easily become a bottleneck because transcoding poses a considerable computational burden. Thus, P2P-based mobile streaming has been actively studied in order to reduce server load and increase service capacity. P2P-based video-on-demand (VoD) mobile streaming is investigated by the authors of [179]. In the system they proposed, the mobile user establishes a one-to-one connection to a proxy peer chosen in the P2P network. Through that proxy peer, the mobile user searches for a pre-encoded video stored somewhere in the P2P network and downloads it. In [43], when a video is downloaded from several source peers, media transcoding is performed by multiple peers to meet the requirements of the destination peer. To combat peer churn, the video quality is adapted by the transcoders based on the feedback from the destination peer. In [110], a fully collaborative P2P streaming system
is proposed. Instead of all users, only a few mobile users pull a video from the video server through base stations. The pulled video is then shared with other neighboring users via a free broadcast channel, such as Wi-Fi or Bluetooth. PatchPeer [66] also combines the strength of a carrier network (e.g., high availability assurance) and a mobile P2P network (e.g., opportunistic use of resources available at mobile users).
Chapter 3

Live Video Multicast

In the previous chapter, we have reviewed recent advances in peer-to-peer (P2P) video streaming. In this chapter, we present Stanford Peer-to-Peer Multicast (SPPM), a P2P system that achieves low-latency and robustness in live video multicast. In SPPM, a single video server, or source peer, emits a live video stream. The live video stream is delivered to a population of viewers. We denote a viewer’s personal computer or set-top box as peer, as opposed to the client of the client-server model. Unlike the client, the peer not only consumes the video but also provides (relays) the video to other peers by contributing its uplink bandwidth to the system. The contributed uplink bandwidth is used to relay video packets to other peers in the P2P system.

The rest of the chapter is structured as follows. Section 3.2 provides the details of the basic SPPM protocol. In the next two sections, we introduce the active overlay management that allows peers to rearrange their positions in the overlay. In Section 3.3, we explain the un-leeching protocol that actively rearranges peers to overcome clogging, which hampers the system to accept new peers. Section 3.4 presents the tree-height reduction protocol, which allows peers to collaboratively construct and maintain the trees to keep the average tree height low. In Section 3.5, we extend the protocol so that it can be aware of connection restrictions between peers. We then propose a distributed uplink bandwidth allocation scheme that maximizes the number of peers connected to the system in the presence of network address translators.
3.1 Preliminaries

SPPM adopts the push approach for low-latency data dissemination. The push approach requires a persistent route among peers. This route is built by connecting a pair of peers using unicast connections, such as TCP or UDP. Since unicast connections are established on top of the IP layer, the collection of the unicast connections is often called an overlay. Each data path in the overlay starts from the server which is the source of the video packets. The data path can be seen as a chain of peers. On the data path, all intermediate peers act as both parents and children; peers that relay video packets are the parents of the peers that receive the packets. Peers that receive video packets are the children of the peers that relay the packets. When a parent peer can relay packets to more than one child peer, then the path naturally evolves from a chain to a tree structure, with the server being the root and peers being intermediate nodes and bottom nodes. Note that the desired distribution structure becomes a spanning tree because no packet needs to visit the same peer more than once.

3.1.1 Peer Characteristics

In a P2P streaming system, peer characteristics have a strong impact on the system because the peers contribute the resources to the system in the form of relaying capacity. In this section, we investigate the primary peer characteristics, which are closely related to the system performance.

Network Access Bandwidth

Peers connect to the Internet via diverse access networks including the Ethernet, WiFi (802.11), DSL, cable modem, or dial-up modem. Each access technology exhibits heterogeneous characteristics as to average/maximum transmission bitrate, latency, packet loss rate, and the amount of jitter. Among these factors, we focus only on the average transmission bitrate. Each peer’s network connection is abstracted by specifying its uplink bandwidth (i.e., its average uplink transmission bitrate) and downlink bandwidth (i.e., its average downlink transmission bitrate).
Connectivity

A peer needs to establish a connection to another peer in order to relay packets or initiate the forwarding of video packets. However, the presence of network address translators (NATs) and firewalls makes such connectivity difficult, or even impossible in some situations (See Section 2.1). In Section 3.5, we analyze the implications of these limitations and discuss the effectiveness of establishing a server-assisted peer-to-peer connection as well as a distributed algorithm in order to enhance peer-to-peer connectivity.

Lifetime

We have seen that peers in the P2P systems play the role of a server by serving other peers. Unlike typical servers, however, peers are unreliable. Peers arrive at the system anytime (a viewer connects to a session to watch a video) and depart the system at their own will (a viewer quits a session). In some cases, peers may get disconnected from the Internet unexpectedly due to a network failure. A peer’s lifetime is defined as the time during which a peer is being connected to a session. The peer lifetime, or, inversely, peer churn rate, significantly affects the frequency of video disruption because a peer’s departure affects the video flow to peers’ downstream.

Heterogeneous Devices

In this work, we consider only video encoded with an H.264/AVC video codec. We do not consider video encoded with Scalable Video Coding (SVC), Multiple Description Coding (MDC), or layered coding. As a result, peers generally watch a video of the same quality, the same resolution, and the same frame rate. If we incorporate a broader category of devices as peers, such as Personal Digital Assistant (PDAs) or mobile phones, peers differ not only for their access network speed, but also for their display size, the amount of main memory, processor speed, and media capability. In typical client/server streaming systems, live media adaptation is performed at the servers to meet the requirements of heterogeneous users. We discuss how to adapt video for heterogenous peers in Chapter 5.
3.1.2 Single Tree versus Multiple Complementary Trees

Single Multicast Tree

Although a single multicast tree can span all participating peers and be used to distribute video packets to them, it has several weaknesses listed below, which we overcome by multiple complementary trees.

1. **Single point of failure**: consider a spanning tree where the source peer is the root of the multicast tree. Peers that have child peers are intermediate nodes and peers with no child peers are bottom nodes in the tree. When an intermediate node (peer) departs, the nodes below the departing node are disconnected from the tree. This prevents the disconnected nodes from receiving video packets originated from the root node. Since each peer has only one parent peer in the system with a single multicast tree, each parent (intermediate) peer is a single point of failure to its child peers.

2. **Under-utilization of peer uplink capacity**: peers may contribute different uplink capacity to the system. For a simple illustration, suppose that there are two different groups of peers, Class A and Class B. Peers in Class A have the uplink capacity of 1.5R, and peers in Class B have the uplink capacity of 0.5R, where R is the bitrate of the video. When a single tree is constructed among peers, each peer has a single parent that relays the video at the rate of R. Therefore, Class A peers can have up to one child peer. Class B peers, on the other hand, cannot have any child peers because their uplink capacity is lower than R. The under-utilization of the peers’ uplink capacity is due to the coarse granularity of the minimum bitrate required for relaying the video.

3. **Large end-to-end transmission delay**: An end-to-end transmission delay for a peer is defined as a time from when a video packet is generated at the source to when it arrives at the peer. The worst end-to-end transmission delay is the maximum of delays that packets experience while traveling from the source to each peer in the same session. As an illustration, suppose that N peers have identical uplink bandwidth of R, where R is the video bitrate. The only overlay that can span all N peers is a chain of peers, or a tree with out-degree 1 (See Figure 3.9). In this particular case, the
peer located at the end of the chain experiences the end-to-end delay that increases linearly with respect to the number of peers.

**Multiple Complementary Trees**

To overcome the weaknesses of the single multicast tree, we build a set of complementary multicast trees. The source peer divides its video stream into several sub-streams and distributes them down each multicast tree. Peers join all the complementary trees by connecting to a parent peer of each tree. When one of the parents of a peer departs, the peer can still receive the other sub-streams from the remaining parents, thereby mitigating the impact of peer churn. Multiple complementary trees also allow finer granularity in the utilization of the peers' uplink bandwidth. For instance, with two trees, peers with $1.5R$ and peers with $0.5R$ can fully contribute their relaying capacity because the sub-stream bitrate of each tree is reduced to $0.5R$ by dividing the video stream into two disjointed sets of video packets. As we will see in Section 3.4, multiple complementary trees can also achieve lower worst-case end-to-end transmission delay than a single tree.

**3.2 SPPM – Basic Protocol**

SPPM consists of a source peer and a set of ordinary peers. SPPM organizes peers in an overlay of multiple complementary trees. Every tree is rooted at the source peer. Fig. 3.1 illustrates an example overlay in which two complementary trees are constructed among a small group of peers. The overview of the basic SPPM protocol is provided below. Detailed basic protocol descriptions can be found in [159]. For the architecture of the real-time implementation of SPPM, see [27]. In [163, 165, 164], the protocol is evaluated with respect to control overhead, video quality, different parent selection schemes, and packet retransmissions.
Figure 3.1: A peer-to-peer overlay consisting of two complementary distribution trees. The encoded video is packetized and split into disjoint substreams, one for each distribution tree.

### 3.2.1 Source Peer

The media stream, originating from the source peer, is packetized and distributed on different trees in such a way that there are no duplicate packets across the trees. Once video packets arrive at those peers directly connected to the source peer, the packets are then distributed to the other peers in the system from peer to peer. The source peer also keeps track of participating peers’ information, such as their IP address, port number, and NAT type.

### 3.2.2 Join Procedure

Peers subscribe to every tree in order to receive the media stream contiguously. As peers join the system, the trees are incrementally constructed in a distributed manner. When a new peer contacts the video source, the video source replies with session information, such as the number of multicast trees and the video bitrate. In SPPM,
peers are allowed to have child peers in only one of the trees. The tree in which a peer is allowed to have child nodes is called the peer’s **fertile tree**. To that peer, the other trees are **sterile trees**. This restriction allows the trees to achieve lower height in terms of the average number of logical hops to the source. Since peers may experience video disruption when one of the ancestors (intermediate peers along the data path to the video source) fails, low-height trees can reduce potential video disruptions due to peer failure. A similar restriction in tree construction is found in [137] where depth-two trees are built.

Suppose that a new peer \( X \) initiates the join procedure. \( X \) is assigned a fertile tree by the source peer when it contacts the video source. \( X \) also receives a list of parent candidates, randomly chosen from the table of participating peers the video source maintains. Based on the probe replies from the parent candidates, \( X \) learns about their current status and selects its parents for each tree. The parent selection scheme depends on the tree type. For the sterile tree(s), it selects parent peers with the minimum number of hops to the source (logical hops in the overlay). For its fertile tree, on the other hand, \( X \) attempts to select a position where it can be placed closer to \( S \) than to any other sterile nodes based on the probe replies it receives. (A node is called a “fertile” node in its fertile tree, whereas it is called a “sterile” node in its sterile tree.) When \( X \) finds a sterile node whose distance to \( S \) is equal to or smaller than that of at least one fertile peer, \( X \) swaps its position with the sterile node using peer swapping if necessary. Peer swapping, which will be detailed in Section 3.3.1, allows peers to switch their positions in the overlay. Peer swapping may be required since trees are incrementally built in a distributed manner by peers, which have limited knowledge about the overlay. Peers’ knowledge is limited because they learn about the overlay by probing a small fraction of the peers, and hence get only a limited view of the overlay. If all the probed sterile nodes are more distant from \( S \) than from all the probed fertile nodes, \( X \) connects to the fertile node closest to \( S \). Algorithm 1 summarizes how a new node selects its parent node in the fertile tree.

Once the selected candidate parent accepts the attachment request, a data connection is established between the parent and the new peer. After data transmission starts, each child peer periodically sends *Hello* messages to its parents.
Algorithm 1 Parent selection for the fertile tree

\( SP := \) Sterile peer, \( FP := \) Fertile peer

\( SPdepth_{\text{min}} \leftarrow \infty \)

\( FPdepth_{\text{max}} \leftarrow -\infty \)

\( AvailFPdepth_{\text{min}} \leftarrow \infty \)

\( L = \{P_1, \ldots, P_n\}, \) \( L \) is the set of \( n \) replied peers

if \( L = \emptyset \) then
  return
end if

Choose the next peer \( P_i \in L \)

if \( P_i \) is \( FP \) then
  if \( \text{Depth}(P_i) > FPdepth_{\text{max}} \) then
    chosenFP \( \leftarrow P_i \)
    \( FPdepth_{\text{max}} \leftarrow \text{Depth}(P_i) \)
    if \( \text{Depth}(P_i) < AvailFPdepth_{\text{min}} \) and \( \text{AvailBW}(P_i) > R_{\text{tree}} \) then
      chosenAvailFP \( \leftarrow P_i \)
      \( AvailFPdepth_{\text{min}} \leftarrow \text{Depth}(P_i) \)
    end if
  end if
else
  if \( \text{Depth}(P_i) < SPdepth_{\text{min}} \) then
    chosenSP \( \leftarrow P_i \)
    \( SPdepth_{\text{min}} \leftarrow \text{Depth}(P_i) \)
  end if
end if

if \( FPdepth_{\text{max}} > SPdepth_{\text{min}} \) then
  ChosenPeer \( \leftarrow \) chosenSP
else
  ChosenPeer \( \leftarrow \) chosenAvailFP
end if

return \( \) ChosenPeer

3.2.3 Rejoin Procedure

When a parent peer leaves the system ungracefully, its children notice that neither video packets nor a response to \textit{Hello} messages arrive. Each abandoned child then initiates a procedure to connect to a different parent node in the tree. During this rejoin procedure, an abandoned child may try to rejoin one of its descendants. This
CHAPTER 3. LIVE VIDEO MULTICAST

will create a loop that may cause its subtree to receive duplicate packets or no new video packets. To prevent a loop, peers keep the list of ancestors on the path to the video source and detect a potential loop. At the same time, the abandoned child asks the parents of its other trees to transmit missing packets that belong to the disconnected tree(s) (See [164] for more details regarding the video- and network-aware retransmission used in SPPM).

When a child peer leaves the system ungracefully, its parents detect the departure by observing consecutive missing Hello messages, and they stop forwarding video to the child. Child failures cause temporary waste of the peer uplink bandwidth due to relaying video to the peers that left the system. Since a false child failure detection may trigger unnecessary video disruptions and rejoin procedures at the affected child and its descendants, a longer timer is used to confirm failure compared to parent failure detection. When a child fails or departs with notice, its parent stops forwarding video packets by removing it from its forwarding table.

3.3 Overlay Un-leeching Protocol

Since a video stream is disseminated through the overlay, the overlay has a strong influence on the system performance, such as the end-to-end transmission delay, system throughput, and video disruption. During a video session, the overlay incrementally grows as peers join the system. When peers leave the system, the overlay reconnects peers that are disconnected from the overlay due to the leaving peers. If the overlay adds a newly arriving peer to its existing structure without any structural change and reacts only to peer departure, the overlay is said to be passively maintained. For SPPM, we propose the active overlay management, which allows peers to proactively reconfigure the overlay structure.

In this section, as part of the active overlay management, the un-leeching protocol is introduced. We define a leech as a peer that does not contribute to a system at all, or contributes uplink bandwidth less than $R$, but that nevertheless receives the video at the bitrate $R$. A data path is defined as a logical route from the source to a node in the tree. When leeches block all the data paths in the overlay, no new peers can join
the system. A P2P system is said to suffer from clogging when it cannot accept any more peers. In Figure 3.2, instances when the system can and cannot accommodate new peers are contrasted.

Figure 3.2: An illustration of system clogging. Class A peer can have up to two child peers. Class B peers cannot have any child peers. Left: A system with available bandwidth. Right: A system experiencing clogging. All the paths from the tree root are blocked by Class B peers.

3.3.1 Un-leeching Using Peer Swapping

Peer swapping is a protocol used to switch the position of two adjacent peers as parent-child relationship in the overlay. The peer swapping protocol allows the system to overcome the disadvantage due to unfavorable peer join sequences. Suppose that $S$ is the video source emitting video of bitrate $R$. Assume that Peer $L$ and Peer $H$ have $R_L < R$ and $R_H \geq R$, respectively, where $R_i$ is the uplink bandwidth of Peer $i$. Suppose that $H$ wants to join the overlay in Figure 3.3 (a). Since $L$ is clogging the data path, $H$ cannot join the system. However, in Figure 3.3 (b), $L$ can join the overlay by connecting to $H$. Peer swapping transforms the overlay from Figure 3.3 (a) to (b), so that the system can scale by un-leeching the overlay.

In SPPM, new peers detect and resolve clogging in a distributed manner. When a new peer contacts the source peer, it receives a list of parent candidates. If the
peer fails to find any parent candidate with available bandwidth, it concludes that the system is clogged. If the peer is a leech, it is not allowed to join the system. If not, it selects the leech with the shortest path to the source among those that have responded to the probing. Then, the peer asks the leech’s parent to swap the leech’s position with itself. After the leech’s parent accepts the peer as a child, the peer accepts the leech as its child. After the peer becomes part of the network, the system’s available bandwidth increases by $R_C - R$ ($R_C$ is Peer C’s uplink bandwidth). See Appendix A.1.1 for more details.

### 3.3.2 False Detection of Clogging

Since probing random peers by a joining peer provides only partial information about the available resources, a false detection may occur when none of the available peers in the system are discovered. The consequences of false detection are insignificant for peers that are not leeches. However, leeches are not allowed to join the system in instances of false detection. In this section, we analyze the impact of false detection for leeches. When a new peer joins, it randomly probes $r$ peers out of $N_p$ peers. Suppose that $p \times N_p$ out of $N_p$ peers are available as parents. The probability that the peer finds no available peers is expressed as

$$P(\text{no available peer found}) = \frac{(1-p)^{N_p}}{r^{N_p}}. \quad (3.1)$$
In Fig. 3.4, the rate of false detection is equal to the rejection rate of leeches, with 5% of peers in the system having available bandwidth at the time of probing. Regardless of the group size, more probes decrease rejection rate. When the group size is reasonably large (>300), probing 15 random peers results in 45% of false detection. If an additional 15 random peers are probed, the false detection rate drops to 20%. Although the false detection rate does not drop to 0%, leeches discover available peers by probing a small fraction of the peers in the system quickly.

![Figure 3.4: Probability of false clogging detection with a different number of probes. 5% of peers in the system have available bandwidth at the time of probing.](image)

### 3.3.3 Experimental Results

We evaluated the performance of the un-leeching protocol in ns-2 [13]. In the first experiment, 50 peers with $1.5R$ of uplink bandwidth (Class A) and 50 peers with $0.5R$ of uplink bandwidth (Class B) joined the system (Appendix B presents a theoretical analysis on the configuration used in this experiment). Peers join the system at an arbitrary time, independently of each other. They remain in the system until the video session is over in order to study the effect of the incremental construction of trees with sequential peer joins.
Table 3.1: The percentage of peers connected to all trees. Each row is obtained with the same peer join sequence. (NPS: no peer swapping, PS: peer swapping)

<table>
<thead>
<tr>
<th>Run</th>
<th>1 Tree</th>
<th>2 Trees</th>
<th>4 Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NPS</td>
<td>PS</td>
<td>NPS</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>52</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>52</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>52</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>52</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>52</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>Average</td>
<td>6</td>
<td>52</td>
<td>59.67</td>
</tr>
</tbody>
</table>

Table 3.1 shows the experimental results with randomly generated peer join sequences. We selected representative runs to allow for one-on-one performance comparison. The first column is the index of a peer join sequence. The following columns correspond to the results for 1, 2, and 4 complementary trees, respectively. For each simulation run, we counted peers connected to all trees. For each case, experiments with and without peer swapping were performed. The results demonstrate that without peer swapping, the system throughput largely depends on the peer join sequence.

However, a system with an actively managed overlay manages to connect all peers, except for the system with a single tree. In the system with a single tree, the system throughput is only 52R, which is the aggregate of the uplink bandwidths contributed by the Class A peers, plus the source uplink bandwidth. This limitation arises because Class B peers’ uplink bandwidth is below the tree bitrate R and the Class A peer can only contribute up to 1R. This limitation is resolved with the 2 and 4 trees because all peers can fully contribute their resources to the system.

Fig. 3.5 shows the clogging probability for a group of N = 50 and N = 100 peers for the system of 4 trees. A group of 50 peers consisted of 25 Class A peers with
1.5R and 25 Class B peers with 0.5R. For a group of 100 peers, we used the same configuration used in the earlier setting, which is 50 Class A peers with 1.5R and 50 Class B peers with 0.5R. Fig. 3.5 also shows the fraction of peers connected to the overlay when the un-leeching is not performed. Clearly, higher server capacity causes more peers to connect to the overlay (See Appendix B). Nevertheless, some peers fail to join the system. We found that the un-leeching protocol always allowed all peers to join the system at the end of each simulation run.

![Graph showing clogging probability and fraction of connected peers](image)

**Figure 3.5:** The left y-axis shows the clogging probability. The right y-axis shows the fraction of peers connected to the system when the un-leeching protocol is not employed. All peers connected to the system when the un-leeching protocol was used.

In the second experiment, we applied the uplink bandwidth distribution shown in Table 3.2 (See Fig. D.8 for details). The system built 4 trees and the video bitrate was set to achieve the over-provisioning factor ranging from 0.95 to 1.15. The over-provisioning factor is defined as the ratio of the average of the peer uplink bandwidth distribution to the video bitrate and it indicates how much uplink over-provisioning is contributed from peers. The resulting video bitrate ranged from about 470 kbps to 580 kbps. The simulation results for the over provisioning factor of 1.05 are shown in Fig. 3.6. The figure shows that system clogging occurred less often when the server
Table 3.2: Peer uplink bandwidth distribution based on the measurements. (The data are based on Fig. D.8 in Appendix D)

<table>
<thead>
<tr>
<th>Uplink bandwidth [bps]</th>
<th>400k</th>
<th>700k</th>
<th>1000k</th>
<th>1300k</th>
<th>1500k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction</td>
<td>43%</td>
<td>32%</td>
<td>12%</td>
<td>7%</td>
<td>6%</td>
</tr>
</tbody>
</table>

accepted more direct children. As the group size increased, the system was more often clogged, making un-leeching more important. When a group size increases, the available capacity at the end of the session is likely to be higher if no clogging occurs. However, its available capacity is more likely to go below zero at least once when more peers join the system (See Appendix B). The same trend was observed in Fig. 3.5 of the previous experiment. We observed that the un-leeching protocol allowed all peers to join the system at the end of the simulation runs.

Fig. 3.7 shows the effect of the over-provisioning of peer uplink bandwidths on clogging. The number of direct children at the server was set to 4. The $x$-axis represents the over-provisioning factor. The average of actual peer uplink bandwidths varied in each simulation run. When the average peer uplink bandwidth was around or below the video bitrate, the clogging probability was very high. As the video bitrate decreased, the over-provisioning increased and the clogging probability decreased accordingly. This implies that when the system is not sufficiently provided with peer uplink resources, the system is likely to experience clogging at some point as peers join the system. This experiment shows that the system benefits most from the un-leeching protocol when the system operates with the over-provisioning factor below 1.15 (the average uplink bandwidth exceeding the video bitrate by 15% or less). We also found that some leeches (peers with 400 kbps uplink bandwidth) took much time to join when the over-provisioning factor was low, sometimes failing to join when the over-provisioning factor was close to or below 1. Nevertheless, all peers with uplink bandwidth 700 kbps or above joined the system by the un-leeching protocol.
Figure 3.6: Probability of system clogging. Peers with their uplink bandwidths following the distribution in Table 3.2 arrive in the system. The over-provisioning ratio was 1.05. The clogging probability decreases as the server provides more bandwidth.

Figure 3.7: Probability of system clogging. Peers with their uplink bandwidths based on the distribution in Table 3.2 arrive in the system. The server accepts 4 direct children. The clogging probability decreases as the video bitrate decreases.
3.4 Tree-Height Reduction Protocol

In Section 3.2.2, we discussed the parent selection algorithm used by newly arriving peers. According to the algorithm, a new peer selects the peer closest to the root (in terms of the number of logical hops) among parent candidates. We define the average tree height as the average number of logical hops from all the nodes in the tree to the root. Then, the parent selection algorithm ensures that the average tree height is kept as low as possible as the tree grows. However, the parent selection process is performed only when peers join the system, thereby the existing peers being oblivious of the overlay changes incurred by peer dynamics. We also found that the average tree height can be further reduced if parents are allowed to rearrange existing connections. We propose the tree-height reduction protocol to alleviate such shortcomings. Unlike the parent selection used in Section 3.2.2, the tree-height reduction protocol ensures that the tree adapts to the structural changes it experiences due to peer arrival or departure. It also renders the tree height less sensitive to the peer arrival sequence. The tree-height reduction protocol is a part of the active overlay management as it proactively restructures the overlay.

3.4.1 Child Promotion

The tree-height reduction protocol uses child promotion to reduce the average tree height. With child promotion, nodes with a large out-degree are moved closer to the root of the tree. Child promotion is initiated and conducted in a distributed manner. Peers, as parents, determine if they need to conduct child promotion. When child promotions are iteratively performed at the nodes in the tree, more nodes are placed closer to the tree, shaping the tree into a fatter, shorter tree. Peer swapping, introduced in the previous section, is employed in promoting a child by one level. Unlike in un-leeching, peer swapping for child promotion switches two nodes that are already part of the overlay. Note that the child promotion based on the peer swapping presented in this section does not always guarantee a globally optimal tree, where no peers at the higher level have a smaller out-degree than the peers at the lower level. Nevertheless, child promotion works effectively in practice.
Figure 3.8 illustrates a child promotion in which Peer X promotes its child Peer Y by swapping its position with Peer Y. In SPPM, peers regularly exchange their local tree structure information by Hello messages. Peer X initiates the swapping when one of its child nodes reports a higher out-degree than its own out-degree. If there is more than one child node that meets this triggering condition, Peer X chooses the child node with the largest out-degree. If there is a tie, the child with the largest uplink bandwidth is chosen. Suppose that Peer Y is Peer X’s child with the largest out-degree, which is higher than Peer X’s out-degree. If Peer Y has no more available bandwidth, Peer Y selects one of its current child nodes, depicted as Peer Z in Figure 3.8. Otherwise, Peer Z is not selected and is henceforth ignored. Peer Z connects to Peer X as a child by taking over the bandwidth previously occupied by Peer Y. Note that the rest of the child nodes and their descendant nodes are kept intact in order to minimize the change in the overlay. As in the peer swapping for un-leeching the overlay, the peers that change their parent nodes, Peer X, Y, and Z, enter the temporary lock state until the swap is complete. Peers in the lock state prevent themselves from being involved with other concurrent overlay rearrangement. The lock state expires when no reply arrives within a deadline, which allows sufficient time for control packets to travel among the peers. Since peer swapping involves two or three closed-loop communications (one between Peer X and Y, another between Peer Y and G, and the other (if any) between Peer Z and X), peer swapping is completed depending on how far the peers are physically distant from each other\(^1\) (See Appendix A.1.2 for the protocol flow in time).

Once the swap is complete, Peer G delivers video packets to Peer Y. Peer Y receives video from both Peer X and G until Peer X stops forwarding packets. Peer X may receive video from both Peer Y and G. Nevertheless, no duplicate packets arrive at Peer X because Peer G stops sending packets to Peer X after Peer G accepts Peer Y. Peer X avoids creating forwarding loops by not transmitting video packets back to Peer Y. Even when duplicate packets arrive at Peer X or Y due to retransmissions, only one copy is forwarded to their other children.

\(^1\)In ns-2 simulations, the average time for peer swapping was around 200 ms, which was 2.5 times the largest round-trip time between peers in the network used for the experiments.
3.4.2 Modeling Worst End-to-End Delay

In this section, we examine the effect of the number of trees on the end-to-end transmission delay. To capture the relationship between the number of trees and the worst end-to-end transmission delay, we consider a configuration that allows for the derivation of simple closed-form mathematical expressions. In this configuration, each peer contributes the same amount of uplink bandwidth $R$, which is identical to the video bitrate $R$. In two extremes, we can construct either a single tree or as many trees as there are peers in the system. With the system building a single tree, the tree has a chain structure, where every intermediate node has out-degree 1. Every parent node in a tree, apart from the source node, is called an intermediate node. With the system building as many trees as peers, the direct child of the source, which is the only intermediate node, has connections to the rest of the peers. Therefore, the intermediate node has out-degree $N - 1$, where $N$ is the number of nodes. The analysis in the following section shows that both extremes exhibit a linear increase in delay with the number of peers, which certainly keeps the streaming system from being scalable with respect to the number of peers. Fig. 3.9 illustrates the two extreme cases.

Now, we model the worst end-to-end transmission delay of the overlay. Let $N_t$
Figure 3.9: Left: a single tree. $S$ denotes the source node. $P_i, i = 1, 2, 3, 4,$ denote regular nodes. Right: $N = 4$ trees are built in the system, where $N$ is the number of nodes except the source. The four nodes appear as the only intermediate node in one of the trees. $S_i, i = 1, 2, 3, 4,$ denote the logical sources that correspond to the same physical source. The trees rooted at each logical source are logically isolated from each other.

denote the number of trees. $N_t$ takes on values from 1 to $N$, where $N$ is the number of peers. We set $N_t$ to the maximum out-degree of nodes. A fraction of the nodes serve as intermediate nodes in each tree, forwarding data from the source to the rest of the nodes. The number of intermediate nodes in a tree is determined by the maximum out-degree and the number of peers in the system. This in turn determines the tree height, which is defined as the maximum number of hops from the source to a node.

When all the intermediate nodes have $N_t$ child nodes, a tree of maximum depth $h$ can contain up to $\sum_{i=1}^{h} N_t^{i-1}$ nodes. Then the number of nodes $N$ and tree height $h$ meet the following condition

$$\sum_{i=1}^{h-1} N_t^{i-1} < N \leq \sum_{i=1}^{h} N_t^{i-1}. \quad (3.2)$$

In the case of $N_t = 1$, we have $h = N$ (a chain of nodes is built). For $N_t > 1$, we manipulate the second inequality, which yields

$$h \geq \log_{N_t} (N(N_t - 1) + 1). \quad (3.3)$$
The first inequality in (3.2) is manipulated similarly. Then, together with the condition that \( h \) is a nonnegative integer, \( h \) can be written as

\[
h = \left\lceil \log_{N_t} (N_t(N_t - 1) + 1) \right\rceil.
\]

(3.4)

Note that (3.4) indicates that the tree height decreases when the number of trees \( N_t \) increases. Figure 3.10 illustrates a tree with out-degree four. \( S \) represents the root node, and \( P_1 \) and \( P_2 \) represent intermediate nodes. Peer \( P_h \) is a leaf node with depth \( h \). The end-to-end delay from \( S \) to \( P_h \), which is the worst delay in the system, is denoted as \( T \). The worst end-to-end delay for a system is defined as the maximum of delays that packets experience while traveling from the source to each destination. The delay at each hop is comprised of three components: propagation delay, queueing delay, and transmission delay.

\[\text{Figure 3.10: An example of an out-degree-4 tree (} N_t = 4) . \text{ } P_i \text{ is a peer at depth } i \text{ in the tree.}\]

*Propagation delay* is the time required for a packet to propagate from one peer to another peer. We assume that the propagation delay between any two peers (including \( S \)) is identical, denoted by \( T_p \).

*Queueing delay* occurs because intermediate nodes duplicate packets in order to forward the duplicates to their child nodes. When a packet from \( S \) arrives at \( P_1 \),
it is duplicated to \( N_t \) identical packets for \( P_1 \)'s child nodes. The same duplication occurs at each hop along the data path until \( P_h \) receives the packet. Assuming that intermediate nodes relay packets to the next hop nodes immediately after they receive them, the queueing delay for the last node served by \( P_1 \) is expressed as \( (N_t - 1)L/R \), where \( L \) is the size of a packet in bits and \( R \) is the uplink capacity of the peers. Note that no other traffic is queued up in the intermediate nodes’ queues because each node serves as an intermediate node in only one of the trees. For simplicity, we ignore queueing delays that occur in the network. We also assume that there is no other application traffic that involves peers’ uplink bandwidth usage.

Transmission delay is the amount of time needed to place a packet into the network. The transmission delay of a packet with size \( L \) is \( L/R \) for regular peers and \( L/R_S \) for the source peer (\( R_S \) is the uplink bandwidth of the source peer).

Putting together these three delay components, the worst end-to-end delay \( T \) is given by

\[
T = hT_p + L \left( \frac{1}{R_S} + \frac{(h - 1)N_t}{R} \right),
\]

where \( h \) is obtained from (3.4).

Worst End-to-end Delay Estimated by the Model

Figure 3.11 shows the worst end-to-end delays for systems with 2, 4, 6, 8 and 12 trees predicted by the model presented earlier. The packet size was set to 1500 Bytes and the uplink speed of the source and regular peers was set to 1 Mbps. The propagation delay \( T_p \) was set to 30 ms. In the figure, the worst delay increases logarithmically with the number of peers \( N \). Note that \( N_t \) controls the ratio between the propagation delay along the delivery path and the queueing delay at peer queues. On the contrary, for two extreme cases, \( N_t = 1 \) and \( N_t = N \), we observe that the end-to-end delay increases linearly with the number of peers, also depicted in Figure 3.11. When \( N_t = 1 \), nodes are chained together in the overlay because each peer can have only one child peer. Let \( T_{\text{Chain}} \) denote the worst end-to-end delay for this chain type of overlay. The length of the chain is equal to the number of peers \( N \). Since each
peer has a single child peer, no queueing delay occurs at peer queues. Thus, $T_{\text{Chain}}$, experienced by the peer positioned at the end of the chain, is

$$T_{\text{Chain}} = T_p + \frac{L}{R_S} + (N - 1)\left(T_p + \frac{L}{R}\right). \tag{3.6}$$

When $N_t = N$, each tree is of height 2 with out-degree $N_t - 1$, resulting in the shortest tree with the largest out-degree. Trees employed in the MutualCast system \cite{113,112} are an example of this case. In MutualCast, the media source first sends data packets to its direct child peers for each of the $N$ trees. Then, the peers forward the received packet to the rest of the peers. The worst end-to-end transmission delay $T_{\text{MC}}$ of MutualCast is

$$T_{\text{MC}} = 2T_p + \frac{L}{R_S} + (N - 1)\left(\frac{L}{R}\right). \tag{3.7}$$

These two extreme cases of delay demonstrate that when the number of trees is small, queueing delay incurred at each intermediate peer is small. However, packets experience long propagation delay due to large tree height. When the number of trees is large, the total propagation delay along the path is small. However, packets experience large queueing delay due to the large out-degree.

Figure 3.12 illustrates the relationship between the number of trees (equal to constant out-degree for our special case) and the end-to-end delay. Regardless of the group size, the minimum delay is achieved with around 4 to 8 trees.

### 3.4.3 Experimental Results

We conducted simulations in the ns-2 simulator to measure end-to-end transmission delays. Peers were placed on the randomly chosen edge nodes of the backbone network. The backbone links were sufficiently provisioned. The propagation delay of the network links was set to 5 ms. Peers remained in the system until the video session was over. The 280 kbps video stream was divided into $N_t$ disjoint substreams and
Figure 3.11: Worst end-to-end delay under different number of peers. The number of trees is equal to the out-degree except for degree $N$, for which the number of trees is equal to the number of peers. ($L = 1500$ Bytes, $R_S = R = 1$ Mbps, $T_p = 30$ ms)

Figure 3.12: Worst end-to-end delay under different number of trees. The group size is the number of peers in the system. The number of trees is equal to the out-degree. ($L = 1500$ Bytes, $R_S = R = 1$ Mbps, $T_p = 30$ ms)
delivered through as many complementary multicast trees. The packet size \( L \) was set to 1500 Bytes.

First, we consider the same configuration we assumed for the delay model in Section 3.4.2. All peers including the source peer share the same uplink bandwidth, \( R_S = R = 308 \text{ kbps} \) (10% of uplink bandwidth was over-provisioned to reduce the queuing delay). The average one-to-one delay between peers \( T_P \) was around 30 ms. Peers computed the average end-to-end delay of packets delivered through each tree, respectively. Peers then chose the worst mean end-to-end delay among all the trees. Finally, the mean of the worst end-to-end delay across all the peers was computed over 50 simulations. Figure 3.13 depicts the mean worst end-to-end delay for the system with four trees. The delay grew logarithmically as more peers joined, which matches the results of the analysis in the previous section. Around \( A \) and \( B \) in the figure, however, peers started to experience larger delays (the effect is smaller at Point \( B \) as a relatively smaller fraction of peers are immediately affected by the new layer). These abrupt increases were caused by an additional layer of peers formed in every tree.

Figure 3.14 illustrates the worst end-to-end delays depending on the number of trees and the number of peers, where the number of trees is equal to peers’ out-degree. The worst end-to-end delay for one tree, where peers form a chain, exhibits the largest delay. Depending on the number of trees (or out-degree), the abrupt change in delay occurs with a different number of peers. When the group size exceeds 100, the systems with four to six trees demonstrated the smallest delay. Similar patterns were predicted by the model.

Second, we consider the system configuration where peers fully contribute their heterogeneous uplink bandwidth and peer failures are simulated. The average peer lifetime was set to 120 seconds. The video bitrate was set to 520 kbps. The source uplink bandwidth was set to 572 kbps and the regular peers’ uplink bandwidths were set according to the distribution presented in Table 3.2. For their fertile trees, peers accepted children up to their uplink bandwidths. With peer swapping, peers rearranged their positions in their fertile trees as they accepted children with higher out-degrees.
Figure 3.13: Mean worst end-to-end delay for the system with four trees. Peers start to experience larger delays at Point A and B as peers continue to join the system.

Figure 3.14: Mean worst end-to-end delays for different number of trees and peers. The number of trees is equal to the out-degree.
Fig. 3.15 illustrates the mean worst end-to-end transmission delays at the peers. Simulations with one tree were not feasible because a large fraction of peers were not able to contribute their uplink bandwidths, thereby not reaching sufficient relaying capacity. The lowest delay was achieved when the number of trees was between 4 and 8. As the number of trees exceeded 8, the delay continued to increase. The increase was larger when the group size was bigger. This result echoes our findings based on the model in the earlier section (See Fig. 3.11). Fig. 3.16 shows the average tree height. The number of hops (or intermediate peers) to the tree root was sampled at peers every second. Then the average number of hops was computed over all the samples. It is clear that the average tree height is reduced by increasing the number of trees. Since peer failures at intermediate nodes may result in temporary video disruption and/or retransmission requests, lower tree height is desirable for robust video streaming. However, the decrease in the tree height comes at the expense of the end-to-end transmission delay. As discussed in Section 3.4.2, reducing the tree height requires a significant increase in out-degrees at intermediate nodes, thereby worsening the end-to-end delay. In our experiments, building 4 to 10 trees was a good choice to achieve a trade-off.

3.5 NAT-Aware Bandwidth Allocation

In Section 3.1.1 we saw that NATs or firewalls might present a hurdle for connecting two peers. Inability to connect peers affects the forwarding capacity, thus hampering the system’s scalability. On average, one of six randomly selected pairs of peers fails to establish a connection despite assistance from an external server, such as a STUN server. In this section, SPPM protocol is enhanced to maximize the number of peers connected to the system while peers’ NAT types are taken into account. A P2P system is self-sustainable if the aggregate relaying capacity exceeds the minimum requirement of bandwidth to deliver video to all users. We propose an algorithm that determines how peers allocate their uplink bandwidth to peer types to increase the expected number of peer-to-peer connections successfully established in the system. We first define the self-sustainability for a system, followed by an optimization problem
CHAPTER 3. LIVE VIDEO MULTICAST

Figure 3.15: Mean worst end-to-end delays of trees for different number of trees and peers. Peer average lifetime is 120s. 50 simulations of length 900 seconds are run for each point in the curves.

Figure 3.16: Average tree height in terms of the logical hops from the tree root to all the peers. Peer average lifetime is 120s. 50 simulations of length 900 seconds are run for each point in the curves.
to maximize system self-sustainability. The numerical solution to the optimization problem is presented and the universal solution is studied.

### 3.5.1 Access Type of Peers

A basic technique to improve peer-wise connection is to employ STUN, Session Traversal Utilities for NAT [152], which requires a STUN server working as a rendezvous point. The STUN protocol helps a peer to discover the presence of a network address translator (NAT) and to obtain the public IP address and port number that the NAT has allocated for SPPM’s UDP connections to a remote peer. Since the STUN protocol enables two peers behind a certain type of NATs to set up a UDP connection, it improves the system’s relaying capacity. Table D.1 in the appendix lists peer-to-peer connectivity between different NAT/firewall types of peers. The table shows that the assistance of the STUN server enables more peers to make a direct connection with other peers. However, the following pairs of NAT/firewall types, (1) restricted port NAT ↔ symmetric NAT and (2) symmetric NAT ↔ symmetric NAT, cannot be connected even with the assistance of the STUN server.

Although the peers with full-cone NAT, static IP, static IP with firewall, or restricted IP NAT show different behavior with respect to the needs for the STUN server, they can connect to/from any other peer with server assistance if necessary. Henceforth, we call them Type A. Peers with restricted port NAT are called Type B; Type B peers cannot make a connection with symmetric NAT peers. Finally, peers with symmetric NAT are called Type C; they can only make a connection with Type A peers. Fig. 3.17 depicts the percentage of each NAT/firewall type in Appendix B. Based on Table D.1 and Fig. 3.17, Fig. 3.18 shows that $14.3 + 2.2 + 21.4 + 4.4 = 42.3\%$ of peers account for Type A, $42.4\%$ for Type B, and $15.3\%$ for Type C, respectively. The probability that a pair of randomly selected peers is successful in establishing a direct connection is

$$P(\text{connection establishment}) = 1 - 2P(E_B)P(E_C) - P(E_C)^2$$

where event $E_X$ denotes that a Type X peer is selected. By substituting $P(E_B) =$
0.424 and \( P(E_C) = 0.153 \) into (3.8), we obtain \( P(\text{connection establishment}) \approx 0.847 \). This result indicates that one of six connection attempts would result in failure.

Figure 3.17: NAT/firewall type of peers

Figure 3.18: Access type of peers
3.5.2 Self-Sustainability of the System

In addition to their access type, peers’ uplink bandwidths also affect peer-wise connection. In order for a P2P system to be self-sustainable, the bandwidths contributed by the peers must be sufficient to relay data to every peer. Since there is no central entity in SPPM, peers make a local decision in reserving their bandwidth for each peer type. A peer’s maximum out-degree, or maximum number of outgoing links, equals the maximum number of children that its uplink bandwidth allows for. For each outgoing link, Type A peers select one of the three types according to the probabilities associated with each type. Type B peers select either Type A or B for outgoing links. Type C peers always reserve their bandwidths to Type A peers. Table 3.3 summarizes the probabilities associated with peer types.

Table 3.3: Selection probability of peer type. The left column shows the parent type and the top row shows the child type. $0 \leq \alpha, \beta, \gamma \leq 1$ and $\alpha + \beta \leq 1$.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$1 - \alpha - \beta$</td>
<td>$\beta$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>B</td>
<td>$1 - \gamma$</td>
<td>$\gamma$</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We further assume that peers’ local decisions are independent of each other. Let $D_A$ denote the aggregate of maximum number of out-going links provided by Type A peers. The total number of out-going links reserved by Type A for each type follow the multinominal distribution:

$$P_{A_A, A_B, A_C}(a_A, a_B, a_C) = \begin{cases} \frac{D_A!}{a_A!a_B!a_C!}(1-\alpha-\beta)^{a_A}\beta^{a_B}\alpha^{a_C} & \text{if } a_A+a_B+a_C=D_A \\ 0 & \text{otherwise} \end{cases}$$  (3.9)

where $A_A$, $A_B$, and $A_C$ correspond to Type A out-going links reserved for Type A, B, and C child peers, respectively. The total number of out-going links reserved by Type B peers to Type A and B follow the binomial distribution:
where $B_A$ and $B_B$ correspond to the number of Type $B$ out-going links reserved for Type $A$ and $B$, respectively. $D_B$ denotes the aggregate of maximum out-going links provided by Type $B$ peers. Let $C_A$ denote the total number of Type $C$ out-going links reserved for Type $A$. Since Type $C$ peers offer all of their outgoing links to Type $A$, the total number of Type $C$ peers’ out-going links, $D_C$, equals $C_A$. In order for the system to be self-sustainable, the out-degrees reserved for each type must satisfy the following conditions:

$$A_A + B_A + C_A \geq N_A$$
$$A_B + B_B \geq N_B$$
$$A_C \geq N_C$$

(3.11)

With the constraints $C_A = D_C, A_A = D_A - A_B - A_C$ and $B_A = D_B - B_B$, we rewrite (3.11) as

$$A_B + A_C + B_B \leq D_A + D_B + D_C - N_A$$
$$A_B + B_B \geq N_B$$
$$A_C \geq N_C$$

(3.12)

Note that $A_B$, $A_C$, and $B_B$ are random variables as they reflect how many peers allocate their outgoing links to each type in a distributed manner. Event $S$ is said to occur when the system is self-sustainable, or when the conditions in (3.12) are met. Then, the likelihood of self-sustainability can be expressed as the probability

$$P(S) = P(A_B + A_C + B_B \leq J, A_B + B_B \geq N_B, A_C \geq N_C),$$

(3.13)

where a constant $J = D_A + D_B + D_C - N_A$. 

3.5.3 Maximizing the Likelihood of Self-Sustainability

A P2P system has no control over parameters associated with peers, such as their access type and uplink bandwidth. Nevertheless, the system can control $\alpha$, $\beta$, and $\gamma$ that peers refer to in making a local decision on their bandwidth allocation. In this section, we consider the maximization of the likelihood of the self-sustainability of a P2P system.

**Problem Formulation**

By introducing an indicator function $I(a_B, a_C, b_B)$, we first rewrite (3.13) as

$$I(a_B, a_C, b_B) = \begin{cases} 1 & \text{if } a_C + a_B + b_B \leq J, a_B + b_B \geq N_B, a_C \geq N_C \\ 0 & \text{otherwise} \end{cases}$$

$$P(S) = E\{I(a_B, a_C, b_B)\} = \sum_{a_C + a_B + b_B \leq J, a_B + b_B \geq N_B, a_C \geq N_C} P(a_B, a_C, b_B)$$ (3.14)

where $J = D_A + D_B + D_C - N_A$. $P(a_B, a_C, b_B)$ factors into $P(a_B, a_C)P(b_B)$ for any $a_B, a_C$, and $b_B$. $P(a_B, a_C)$ can be written as $P(a_B, a_C) = P(D_A - a_B - a_C, a_B, a_C)$ in (3.9). $P(b_B)$ can be written as $P(b_B) = P(D_B - b_B, b_B)$ in (3.10). Then, the optimal $\alpha, \beta, \gamma$ is derived by solving the following constrained optimization problem:

$$\text{maximize } \sum_{a_C + a_B + b_B \leq J, a_B + b_B \geq N_B, a_C \geq N_C} F_1(a_B, a_C|\alpha, \beta)F_2(b_B|\gamma)$$ (3.15)

subject to

$$0 \leq \alpha \leq 1$$

$$0 \leq \beta \leq 1$$

$$\alpha + \beta \leq 1$$

$$0 \leq \gamma \leq 1$$
where \( F_1(a_B, a_C|\alpha, \beta) = \frac{D_A!}{(D_A-a_B-a_C)!a_B!a_C!} (1-\alpha-\beta)^{D_A-a_B-a_C} \beta^a_B \alpha^a_C \) and \( F_2(b_B|\gamma) = \frac{D_B!}{(D_B-b_B)!b_B!} (1-\gamma)^{D_B-b_B} \gamma^b_B \).

**Numerical Solutions**

Table 3.4 summarizes the peer statistics used in the illustrative cases. The peer statistics, including the peer type distribution and per-type average out-degree distribution, were based on the Internet measurements (See Appendix D). While keeping the relative ratio among the types, the average out-degree (or over-provisioning factor) was adjusted to be 1.1 for Case 1, and 1.2 for Case 2, respectively. The average out-degree is determined by the video bitrate and peer uplink bandwidths.

The objective function in (3.15) is non-convex. To solve the problem numerically, we performed an exhaustive search over a three-dimensional grid of points with the granularity of 0.01 in the \( \alpha, \beta, \) and \( \gamma \) space, \( 0 \leq \alpha, \beta, \gamma \leq 1 \) and \( \alpha + \beta \leq 1 \). For every point in the grid we computed the objective function in (3.15) and searched for its maximum value. Since the objective function is smooth with respect to \( \alpha, \beta, \) and \( \gamma, \) the error of the solution is between \( \pm 0.01 \) for each dimension of \( (\alpha, \beta, \gamma) \). (Note that one can reduce the error by applying an iterative local search within a smaller grid.)

![Table 3.4: Peer characteristics used for Cases 1 and 2.](image)

<table>
<thead>
<tr>
<th></th>
<th>( N )</th>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d_A )</th>
<th>( O_R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>300</td>
<td>0.423</td>
<td>0.424</td>
<td>0.153</td>
<td>1.35</td>
<td>1.08</td>
</tr>
<tr>
<td>Case 2</td>
<td>300</td>
<td>0.423</td>
<td>0.424</td>
<td>0.153</td>
<td>1.47</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Fig. 3.19 shows the self-sustainability \( P(S) \) over the \( (\alpha, \beta) \) and \( (\beta, \gamma) \) spaces. For both cases, two local maxima were found. In Case 1, the two local maxima were \( P(S) = 0.903 \) at \( \alpha = 0.33, \beta = 0.15, \gamma = 1 \), and \( P(S) = 0.880 \) at \( \alpha = 0.33, \beta = 0.67, \gamma = 0.22 \). High \( P(S) \) values were observed along the lines on the \( (\beta, \gamma) \) space. Fig. 3.19 shows that the system with a more out-degree achieves higher self-sustainability.
Figure 3.19: The likelihood of self-sustainability over \((\alpha, \beta)\) and \((\beta, \gamma)\). Case 1: (a) and (b). Case 2: (c) and (d).

Table 3.5 shows the maxima and their corresponding \(P(S)\) values. In both cases, \(\gamma\) was close to 1 and \(\beta\) was close to 0. These results suggest that for these cases, high self-sustainability is achieved when Type B peers accept only other Type B peers as children and Type A peers accept only Type B peers that could not find a Type B parent. We also investigated the equal-allocation scheme, which equally divides out-going links to each type. With the equal-allocation scheme \((\alpha = 1/3, \beta = 1/3, \gamma = 1/2)\), Type A out-going links are equally divided to each type and Type B out-going links are equally divided to Type A and B. In Table 3.5, where \(P_E(S)\) denotes the self sustainability likelihood under the equal-allocation scheme, we see
that the equal-allocation scheme performs poorly when the over-provisioning of peer uplink bandwidths is small.

Table 3.5: Self-sustainability comparison of the equal-allocation $P_E(S)$ and the numerical solution $P_N(S)$ of (3.15).

<table>
<thead>
<tr>
<th></th>
<th>$P_E(S)$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
<th>$P_N(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.0199</td>
<td>0.33</td>
<td>0.15</td>
<td>1.00</td>
<td>0.9037</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.2299</td>
<td>0.37</td>
<td>0.11</td>
<td>1.00</td>
<td>0.9996</td>
</tr>
</tbody>
</table>

Although we can always find the numerical solution by solving (3.15), it is more practical to employ the universal solution, a fixed set of $(\alpha, \beta, \gamma)$ that generally achieves high self-sustainability likelihood. At the same time, by increasing the number of peer maximum outdegrees we render $P(S)$ to be less sensitive to the deviation of the universal solution from the optimal solution. To increase the number of peer maximum outdegrees, the system either reduces the video bitrate or increases the number of trees increases. To find the universal solution, a system of 100 peers was considered. First, 1000 different populations of 100 peers were randomly generated according to the trinomial distribution $P(E_A) = 0.423$, $P(E_B) = 0.424$, and $P(E_C) = 0.153$. Then, peer maximum out-degrees were computed according to the over-provisioning factor $O_R$ and Table 3.4. For $O_R$, which is the ratio of the average peer uplink bandwidth to the video bitrate, we considered 1.05, 1.10, 1.15, and 1.20.

Fig. 3.20 shows the relation of $N_T$ and $O_R$ on the self-sustainability $P(S)$ of a system of 100 peers. As $O_R$ increases, the system is supplied with more bandwidth, thereby achieving higher $P(S)$. The increase in $N_T$ also achieves higher $P(S)$ for a couple of reasons–a higher $N_T$ allows for not only the better utilization of peer uplink bandwidths, but also the reduction of the variance in the aggregate of out-degrees for peer types. Next, we searched for a fixed $(\alpha, \beta, \gamma)$ that produces the maximum sum of the $P(S)$ values for the configurations while changing the number of trees. In Table 3.6, the numerical solutions for some representative cases are compared with the universal solution $(\alpha_U = 0.34, \beta_U = 0.13, \gamma_U = 1.00)$. Overall, the universal solution achieved high $P(S)$ values close to the $P(S)$ by the optimal solution. The gap between the optimal and universal solution becomes smaller when the peer group
is larger or the over-provisioning factor is higher. This is consistent with the relation between self-sustainability and the aggregate of peer out-degrees observed earlier in Fig. 3.20.

We also observed that the universal solution performed similarly to the optimal solution in most configurations when the over-provisioning factor was over 1.15 and the number of tree was over 8. This suggests that the proposed bandwidth allocation algorithm can be used with a fixed parameter set if the system is operated with sufficient over-provisioning and the moderate or high number of complementary trees.

![Figure 3.20: Self-sustainability of a system of 100 peers. Self-sustainability increases as the number of trees or the over-provisioning increases.](image)

### 3.6 Summary

In this chapter, we proposed the SPPM system for live video streaming in the peer-to-peer context. SPPM builds an overlay of multiple complementary trees in a decentralized fashion. By using the active overlay management we proposed, SPPM proactively maintains the overlay by allowing peers to rearrange their positions in the overlay. When the system suffers from clogging, SPPM allows new peers to locate a leech and join the system by peer swapping. We also studied clogging experimentally
Table 3.6: Numerical solutions \((\alpha_N, \beta_N, \gamma_N)\) and the corresponding \(P_N(S)\) for selected configurations. \(P_U\): self-sustainability likelihood achieved by the universal solution \(\alpha_U = 0.34, \beta_U = 0.13, \gamma_U = 1.00\). \(N_P\): number of peers. \(a\): fraction of Type A, \(b\): fraction of Type B, \(c\): fraction of Type C, \(N_T\): number of trees. \(O_R\): over-provisioning factor.

<table>
<thead>
<tr>
<th>(N_P)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(N_T)</th>
<th>(O_R)</th>
<th>(\alpha_N)</th>
<th>(\beta_N)</th>
<th>(\gamma_N)</th>
<th>(P_N(S))</th>
<th>(P_U(S))</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.46</td>
<td>0.38</td>
<td>0.16</td>
<td>8</td>
<td>1.10</td>
<td>0.32</td>
<td>0.14</td>
<td>1.00</td>
<td>0.9740</td>
<td>0.9690</td>
</tr>
<tr>
<td>100</td>
<td>0.41</td>
<td>0.42</td>
<td>0.17</td>
<td>8</td>
<td>1.12</td>
<td>0.37</td>
<td>0.15</td>
<td>0.99</td>
<td>0.9995</td>
<td>0.9700</td>
</tr>
<tr>
<td>200</td>
<td>0.45</td>
<td>0.42</td>
<td>0.13</td>
<td>6</td>
<td>1.12</td>
<td>0.31</td>
<td>0.12</td>
<td>1.00</td>
<td>0.9997</td>
<td>0.9467</td>
</tr>
<tr>
<td>400</td>
<td>0.40</td>
<td>0.43</td>
<td>0.17</td>
<td>8</td>
<td>1.09</td>
<td>0.35</td>
<td>0.15</td>
<td>1.00</td>
<td>1.0000</td>
<td>0.9980</td>
</tr>
<tr>
<td>500</td>
<td>0.38</td>
<td>0.45</td>
<td>0.17</td>
<td>8</td>
<td>1.08</td>
<td>0.35</td>
<td>0.15</td>
<td>1.00</td>
<td>0.9955</td>
<td>0.8880</td>
</tr>
<tr>
<td>500</td>
<td>0.41</td>
<td>0.41</td>
<td>0.18</td>
<td>8</td>
<td>1.11</td>
<td>0.35</td>
<td>0.12</td>
<td>1.00</td>
<td>1.0000</td>
<td>0.9950</td>
</tr>
<tr>
<td>1000</td>
<td>0.39</td>
<td>0.44</td>
<td>0.17</td>
<td>6</td>
<td>1.08</td>
<td>0.35</td>
<td>0.15</td>
<td>1.00</td>
<td>1.0000</td>
<td>0.9630</td>
</tr>
</tbody>
</table>

and analytically and found that clogging is more likely as the group becomes bigger or the server bandwidth is smaller.

SPPM builds and maintains complementary trees in a way that it achieves smaller end-to-end delay and less video disruptions. We found that there is a trade-off between the physical delay and logical hops, and suggested to use a moderate number of trees (4 to 8) to balance them. Lastly, we studied the effect of NATs on the system relaying capacity and extended SPPM to be NAT-aware. The proposed bandwidth allocation allows peers to locally determine how to allocate their uplink bandwidth for different peer types to maximize self-sustainability.
Chapter 4

Time-Shifted Streaming

In the previous chapter we introduced SPPM, a peer-to-peer live multicast streaming system in which a population of users watch the live video simultaneously. In this chapter, SPPM is extended to allow users to control their playback in the time domain. With playback control, also known as trick mode in video-on-demand systems, peers can individually pause a live video stream, rewind to an arbitrary position (even to contents prior to their time of joining), and fast forward until the latest position of the live video. In the extended SPPM, peers cache the received video in their local buffer. For pause/resume, the locally cached video can be played back at a later time. When a peer rewinds or fast-forwards to content not present in its local buffer, the requested video segment, buffered at some peer(s), is streamed to the peer asynchronously unlike the live stream being multicast to peers. This is called time-shifted streaming. As we will see later, SPPM supports both the live stream and time-shifted streams while keeping the system’s scalability.

The remainder of the chapter is structured as follows. In Section 4.1, we present the extension made to the live multicast SPPM in order to enable time-shifted streaming. In the extended system, peers store past video packets. To keep track of the video contents at peers, the peer database at the source peer is augmented with additional information of peers. The protocol is changed to be able to handle additional fields, such as a video position or a time stamp. Section 4.2 presents the analysis of video availability, which represents how many peers possess video contents of a particular
position. Next, in Section 4.3, we describe fast prefetching to improve video availability. Fast prefetching allows peers to disseminate video faster and the improved video availability is shown to lead to a lower server load. To facilitate video dissemination with fast prefetching, a new parent selection algorithm, called maximum throughput, is proposed and its effects are demonstrated with extensive simulations in Section 4.4.

4.1 System Description

While extending the SPPM system for time-shifted streaming, the fundamental architecture of the system is kept intact so that live video multicast is seamlessly supported. In this section, we describe only the system extensions added to the existing architecture.

4.1.1 Preliminaries

As in live stream multicast, the video server (source peer) starts to stream a live video starting at time 0. Let $V(x)$ denote a video frame associated with position $x$ in time, $x \geq 0$. The live video emanating from the server at time $t$ is thus represented by $V(t)$. A time-shifted stream is represented by $V(t-d)$ with delay $d$, $0 < d \leq t$, at time $t$. Suppose that a new peer or an existing peer requests video $V(x)$ at time $t$. If the peer requests the live video $V(t)$, the peer is called $LS$ peer. If the peer requests a time-shifted stream $V(t-d)$, $0 < d \leq t$, then the peer is called $TS$ peer. For LS peers, the live video is immediately relayed from peer to peer. For TS peers, the live stream must be delayed by time $d$ somewhere in the system, either at the server or at the peers. Delayed streaming, or time-shifted streaming, is achieved by storing past video packets in peers’ buffers while they watch the video\(^1\) and transmitting them when requested.

The time-video plot in Fig. 4.1 illustrates the trajectories of the live stream, Peer 1 (LS peer), and Peer 2 (TS peer). The trajectories indicate the video contents stored

\(^{1}\)A peer shares its storage only when it participates in a session. For a typical two hour-long video stream encoded at 600 kbps, about 540 MB of storage is required. This is a moderate space requirement considering today’s typical personal computers.
CHAPTER 4. TIME-SHIFTED STREAMING

at the peers over time. The vertical axis represents the time stamp of video frames (a time stamp is the time when a video frame is encoded at the server). In Fig. 4.1, Peer 1 requests the live video \( V(t_1) \) at time \( t_1 \). Its video trajectory overlaps the live stream trajectory. At time \( t_2 \), Peer 2 requests the video starting from \( V(x_2) \). Since the requested video is stored at Peer 1, Peer 2 can obtain the video delayed by \( t_2 - x_2 \) from Peer 1.

![Figure 4.1: Video trajectories of live video, Peer 1, and Peer 2. Peer 1, an LS peer, requests the live video. Peer 2, a TS peer, requests the video delayed by \( t_2 - x_2 \).](image)

4.1.2 Peer Database

In the extended SPPM, peers collectively form a distributed video storage. To facilitate the use of the videos stored by peers, locating video among peers is essential. To locate the video of a particular position, the video server maintains a database comprising each peer’s identifier, the time the video is requested, the time stamp of the initial video frame, and the time stamp of the latest video stored in the peer’s buffer. The latest time stamp is estimated by the server based on each peer’s initial video position and request time. Estimation by the server reduces the amount of control traffic between the server and peers by truncating the number of buffer state updates. To prevent the estimation from deviating too far from the actual buffer states, peers may report their buffer state intermittently (see Section 4.3.1).
4.1.3 Video Request and Connection Set-up

When a new peer joins or an existing peer seeks a video of a particular position, it sends a query requesting a certain video position to the server. The server refers to the peer database it maintains. It returns a list of randomly chosen parent candidates whose video buffer may contain the requested (possibly time-shifted) video. The peer then probes each parent candidate to learn about the candidates’ current status, including time stamps of the stored video and available bandwidth. After the peer receives probe replies, it selects and requests for each multicast tree the best parent candidate that will accept it as a child. We discuss parent selection criteria in detail in Section 4.3. Once the parent candidates accept the video request, data connections are established between the parents and the child.

4.1.4 Asynchronous Video Transmission

Connections established for time-shifted streaming are different from connections that constitute the live video multicast trees. In the latter case, packets are immediately relayed to the next hop in the multicast tree, whereas in the former, packets are asynchronously relayed over the time-shifted streaming connection. Multiple complementary trees are the basis of data dissemination in SPPM. As LS peers, TS peers receive packets from multiple parents. Parents asynchronously relay video packets to their TS peers by ensuring that a packet travels along the tree designated to itself.

A parent peer dedicates a streaming window for each TS child peer. Fig. 4.2 shows a streaming window of a TS child that requests video from position $x_1$ at time $t_1$. The lower end of the window, denoted by $W_L$, is the oldest video packet the parent can transmit. Since packets that are past their display time (called the play-out deadline) are not decoded at the child, the value of $W_L$ is set to the minimum of the initial video position in time and of the time stamp of the video frame displayed at the child. The upper end of the window, denoted by $W_U$, corresponds to the time stamp of the latest video packet the parent can send. The value of $W_U$ is selected in such a way that the child’s downlink is not congested by the parent, and the child does not suffer from buffer underflow. The parent regularly searches for packets whose time stamps
are between $W_L$ and $W_U$ in its video buffer. In addition, only packets that belong to the parent’s tree ID are selected to construct the corresponding substream.

To avoid duplicate transmission, the history of transmitted packets is maintained for a finite period of time. Selected packets are marked with their play-out deadline and put into the network output queue. The output queue sorts video packets according to their play-out deadline. Therefore, urgent packets, such as retransmitted packets or live video packets, are sent earlier than packets with a relaxed deadline, such as packets in time-shifted streams.

At a child peer, packets that belong to different substreams may arrive out-of-order. Once received, packets are ordered according to their positions in the video stream before being passed to the video decoder.

![Diagram](image)

**Figure 4.2:** A streaming window for a TS child that requests video from position $x_1$ at time $t_1$. $W_L$ is its lower end and $W_U$ is its upper end. $B$ is the pre-roll delay of play-out at the child. Packets whose time stamps fit in the streaming window are transmitted unless they have been sent already.
4.1.5 Recovery from Disconnect

After video transmission starts, each child peer periodically sends Hello messages to its parents. When a peer leaves the system ungracefully, its parents detect this by observing consecutive missing Hello messages and stop forwarding video to the child. The departing peer’s children notice that neither video packets nor responses to Hello messages arrive. Each abandoned child then initiates a recovery procedure to connect to a different parent node in the tree. During the recovery procedure, LS peers request the latest portion of the video in order to keep end-to-end transmission delay as low as possible. TS peers, on the other hand, request the next position of the video they received if the position was ahead of their playback position. Otherwise, TS children request the playback position of the video.

4.2 Analysis of Video Availability

The availability of time-shifted video contents among peers is closely associated with system scalability because higher video availability lessens the need for making a connection to the server, which leads to server load reduction. Suppose that peers join the system according to a Poisson process $N(t)$ with rate $\lambda$ [42]. Their lifetime follows an exponential distribution with parameter $\mu$ ($1/\mu$ is the average peer lifetime). We assume that the probability that the next peer is an LS peer follows a Bernoulli process with parameter $\alpha$. When a TS peer requests a video, its video access pattern is assumed to be uniform. This assumption allows us to be free from any particular video content and it provides the worst-case study because overlaps between peers’ buffered contents are minimized, which results in minimizing the video availability. Based on this assumption, the position $x$ is selected, at time $t$, uniformly between 0 and $t$. We further assume that a peer always receives the video it requests, by connecting either to another peer or to the video server.

\footnote{A peer lifetime is known to follow a long-tail distribution, such as the Zipf distribution [49, 111, 135]. However, the exponential distribution is still a popular choice for both analysis and simulation study because it reasonably approximates the actual distribution. More importantly, this assumption allows the system analysis to be tractable in that the system behavior can be predicted by observing only a few primary system parameters.}
Let $M(t, x; \lambda, \mu, \alpha)$ denote the number of peers possessing the video at position $x$ at time $t$. To analyze the availability of video contents among peers, we compute the expected value of $M(t, x; \lambda, \mu, \alpha)$, given by

$$E\{M(t, x; \cdot)\} = \sum_{k=1}^{\infty} k \sum_{i=k}^{\infty} \Pr\{M(t, x; \cdot) = k | N(t) = i\} \Pr\{N(t) = i\}, \quad (4.1)$$

where $N(t)$ represents the number of peers that have arrived at the system between 0 and time $t$. Given $N(t) = i$, since $N(t)$ is a Poisson process, peers’ arrival times $S_1, \ldots, S_i$, considered unordered random variables, are distributed independently and uniformly in the interval $[0, t]$ [153]. This allows us to study each peer’s behavior independently regardless of peers’ join sequence. Consider a peer that joins the system between 0 and $t$. Let $p$ denote the probability that the peer is still present and possesses video at position $x$ at time $t$. Then, $E\{M(t, x; \lambda, \mu, \alpha)\}$ in (4.1) can be rewritten as

$$E\{M(t, x; \lambda, \mu, \alpha)\} = \sum_{k=1}^{\infty} k \sum_{i=k}^{\infty} \left( \frac{i}{k} \right) p^k (1 - p)^{i-k} e^{-\lambda t} \frac{(\lambda t)^i}{i!}$$

$$= \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \left( \frac{p}{1-p} \right)^k e^{-\lambda t} \frac{1}{(i-k)!} \sum_{i=k}^{\infty} \frac{(1 - p)^i}{i!} \cdot (\lambda t)^i$$

$$= e^{-\lambda t} \sum_{k=1}^{\infty} \frac{(p\lambda t)^{k-1}}{(k-1)!} p\lambda t = p\lambda t. \quad (4.2)$$

We compute $p$ by considering the cases of TS peer arrivals and LS peer arrivals separately. Suppose that a peer arrives at time $s$, $s \leq t$. If it is a TS peer, then it contains video at position $x$ at time $t$ only when its arrival (or request) time and its requested video position are inside the area surrounded by $(0, 0)$, $(t-x, 0)$, $(t, x)$, and $(x, x)$ on the time-video plot in Fig. 4.3. One can easily check this by drawing a
video trajectory (in parallel with the live stream trajectory) from any point inside the area until time \(t\) and examine whether the trajectory passes or touches the horizontal line drawn from \((0, x)\) to \((t, x)\). Note that the video is not available above the live stream trajectory. Let \(g(s; t, x)\) denote the probability that a TS peer joining at time \(s\) contains the video \(x\) at time \(t\). The value of \(g(s; t, x)\) is 1 when \(0 \leq s \leq (t - x)\), \((t - x)/s\) when \((t - x) < s \leq x\), and \((t - s)/s\) when \(x < s < t\), respectively. Based on the exponential distribution of peer lifetime, the probability that the peer is still present at time \(t\) is \(e^{-\mu(t-s)}\). Then, the probability that a TS peer that contains video at \(x\) at time \(t\), \(l(t, x; \mu)\), is

\[
l(t, x; \mu) = \int_0^t e^{-\mu(t-s)}g(s; t, x)\frac{1}{t}ds
\]

\[
e^{-\mu t} \left\{ \int_0^{t-x} e^{\mu s}ds + (t-x) \int_{t-x}^x e^{\mu s} \frac{ds}{s} + \int_x^t e^{\mu s} \frac{t-s}{s} ds \right\}. \tag{4.3}
\]

When an LS peer arrives at time \(s\), it possesses video at position \(x\) at time \(t\) only when \(s \leq x\). The set of the pairs of the live video position and arrival time is

Figure 4.3: Initial video positions and request times for LS peers (Left) and TS peers (Right) that possess video at position \(x\) at time \(t\). Left: An LS peer whose arrival time is earlier than \(x\) has video at position \(x\) at time \(t\). Right: A TS peer whose request time and initial video position are within the area surrounded by \((0, 0)\), \((t - x, 0)\), \((t, x)\), and \((x, x)\) on the time-video plot has video at position \(x\) at time \(t\).
the line segment from \((0, 0)\) to \((x, x)\) shown on the left side of Fig. 4.3. Then, by incorporating the likelihood of a peer’s departure before time \(t\), the probability that an LS peer that contains video at \(x\) at time \(t\) is expressed as

\[
\int_0^x e^{-\mu(t-s)} \frac{1}{t} ds = \frac{e^{-\mu t}(e^{\mu x} - 1)}{\mu t}.
\]  

(4.4)

Finally, \(p\) can be written as

\[
p = \Pr\{L^c\} \Pr\{x\ \text{is possessed at time } t|L^c\} + \Pr\{L\} \Pr\{x\ \text{is possessed at time } t|L\}
\]

\[
= (1 - \alpha)l(t, x; \mu) + \frac{\alpha e^{-\mu t}(e^{\mu x} - 1)}{\mu t},
\]

(4.5)

where \(L\) is the event that an LS peer joins the system. By inserting (4.5) into (4.2), we rewrite (4.2) as

\[
E\{M(t, x; \lambda, \mu, \alpha)\} = (1 - \alpha)\lambda tl(t, x; \mu) + \frac{\lambda e^{-\mu t}(e^{\mu x} - 1)}{\mu},
\]

(4.6)

where \((1 - \alpha)\lambda tl(t, x; \mu)\) corresponds to the expected number of TS peers that possess video at position \(x\) at time \(t\) and \(\alpha \frac{\lambda}{\mu} e^{-\mu t}(e^{\mu x} - 1)\) corresponds to the expected number of LS peers that possess \(x\) at time \(t\).

In (4.6), the video availability is predicted to increase linearly as the peer’s join rate \(\lambda\) increases. This result demonstrates that the system is self-scalable. The availability of the early portion of the video continues to decrease as \(t \to \infty\), which indicates that time-shifted streams have to be provided by the video server at a later time. In Fig. 4.4, the video availability is predicted by evaluating (4.6) with \(\lambda = 1.125\) joins per second, \(1/\mu = 120\) seconds, and \(\alpha = 0.5\). In the figure, the video near the live stream is available at many peers, the majority of which are LS peers. As time progresses, more video frames become available and the overall video availability decreases. Figs. 4.5 and 4.6 illustrate the expected video availability at 300 and 900 seconds, respectively. In these figures, the contribution from LS peers and TS peers is separately shown. The expected number of TS peers having video at position \(x\) at
time $t$ peaks when $x = t/2$, whereas the expected number of LS peers at time $t$ peaks at the live video position of $x = t$ and its value is $\lim_{t\to x} \alpha \lambda e^{-\mu t} (e^{\mu x} - 1) \approx \alpha \lambda / \mu$. Although the videos stored at either LS or TS peer have high overlaps at time 300s, the overlap dramatically decreases at time 900s. This result implies that at a later time of the session more TS peers have to connect to the server, which was already observed in Fig. 4.4.

![Figure 4.4: Video availability on the time-video plot. The value at $(t, x)$, predicted by the model, represents the expected number of peers available for video at position $x$ at time $t$ ($\lambda = 1.125$ joins per second, $1/\mu = 120$ seconds, and $\alpha = 0.5$).](image)

4.3 Fast Prefetching

4.3.1 Improving Video Availability

In the previous section, we have observed that the video availability decreases over time. Server load is highly affected by video availability because higher video availability allows more peers to obtain video from other peers, thus reducing the number of connections made to the server.
In this section, we propose fast prefetching, which allows peers to download video more rapidly. Fast prefetching exploits the relaxed transfer deadline of time-shifted streams when compared to the live stream. As an input device such as a video camera generates a video signal, the live video is immediately encoded and transmitted to peers. On the other hand, time-shifted streams are the past portion of the video, already encoded and stored somewhere in the system. Fast prefetching also exploits the extra uplink bandwidth of peers. Suppose a TS child receives a time-shifted stream from its parent. When its parent has sufficient uplink bandwidth, the stream can be pushed to the child faster than playback speed. Since fast prefetching facilitates video dissemination among peers, video availability can be improved accordingly. Fast prefetching also improves the video playback quality of TS peers. With fast prefetching, peer buffers grow faster than playback speed, which reduces the urgency for acquiring later packets. When a peer is disconnected due to failure of its parent, it reconnects to the overlay. Since the peer has more video frames to play back until it resumes to receive later video, it experiences statistically less video disruption due to buffer underflow.

Since fast prefetching is based on the relaxed transfer deadline of time-shifted streams, it does not benefit LS peers. When LS peers are disconnected from the
overlay, they often experience buffer underflow because the play-out deadline of a video packet is kept small in live streaming. To compensate for a tight play-out deadline, LS peers connect to peers that can minimize the number of logical hops to the source [27]. Using this criterion, LS peers attempt to reduce disruptions in receiving the video when their parent fails or leaves unexpectedly. The effects of fast prefetching are illustrated in Fig. 4.7. In this illustration, the uplink of the parent is assumed to be twice the video bitrate. $P(t)$ is the video trajectory of the parent. After Child 1 connects to its parent, it prefetches the video faster than the playback speed until Child 2 connects to the same parent. The trajectories of the downloaded video for Child 1 and 2 are depicted as $C_1(t)$ and $C_2(t)$, respectively. If fast prefetching is not employed, Child 1 receives video at the playback speed, which is marked by $C_1(t)$. Since fast prefetching enables peers to download video faster than at the playback speed, the peer database’s estimation on the peers’ buffer state may deviate from the actual buffers. To correct mismatches incurred at the database, peers intermittently report their buffer state to the video server.

![Diagram](image)

Figure 4.7: An example of fast prefetching for TS peers. The parent peer can serve two children concurrently. At time $t$, the excess download at Child 1 due to fast prefetching is $C_1(t) - C_1(t)$. 
4.3.2 Selecting Parents that Maximize Prefetching

LS peers choose their parents according to the number of hops to the server, which is certainly not optimal for TS peers because the video disruption is less likely due to fast prefetching. To maximize prefetching, we propose a new parent selection algorithm for TS peers, which estimates video download for each parent candidate. When Parent Candidate \( i \) is probed by a TS peer that requests the video from position \( x \), the parent candidate reports (1) the download rate \( q_i \) from its current parent, (2) the number of bits cached at Parent Candidate \( i \) and requested by the TS peer (e.g., a part of the requested substream and beyond the position \( x \)), denoted by \( l_i \), and (3) the maximum upload rate it can offer to the TS peer, denoted by \( r_i \). In Fig. 4.8, \( q_i \), \( r_i \), and \( l_i \) reported by Parent Candidate \( i \) are depicted over time. The Y-axis shows the video position in bits. Note that the rates and the buffer amount were adjusted to reflect the number of trees. As each multicast tree delivers a \( 1/k \) fraction of the video bitstream, \( q_i \), \( r_i \), and \( l_i \) were multiplied by \( k \) for the illustration (\( k \) is the number of trees).

After probing the candidates, the TS peer computes the amount of prefetched video that it expects to receive time \( D \) after connecting to each parent candidate: if the peer’s download catches up with its parent’s download before time \( D \) has elapsed, the expected download is \( D \cdot q_i + l_i \). Otherwise, the expected download is \( D \cdot r_i \) (see Fig. 4.8). The value of \( D \) is chosen to be sufficiently small, as \( q_i \) and \( r_i \) may change over time. The best candidate \( k \) is chosen according to the rule

\[
k = \arg \max_i \min (D \cdot q_i + l_i, D \cdot r_i).
\]  

(4.7)

The video server is selected as a parent only when there is no peer possessing the requested video or when no peer that possesses the requested video has available bandwidth. It should also be noted that the server does not participate in fast prefetching in order to minimize its uplink bandwidth usage.
Figure 4.8: Expected download at a child peer after time $D$ when a child peer joining at time $t$ connects to parent candidate $i$. Left: child buffer lags behind parent buffer at time $t + D$. Right: child buffer catches up with parent buffer. ($q_i$: download rate of Parent Candidate $i$ from its parent, $l_i$: buffered video in bits at Parent Candidate $i$, $r_i$: upload rate from Parent Candidate $i$ to the child peer.)

### 4.3.3 Uplink Bandwidth Partitioning

To support fast prefetching, peers partition their available uplink bandwidth to children as follows:

(Step 1) Allocate the tree bitrate $r(= R/k)$ to every LS child. $R$ is the video bitrate and $k$ is the number of trees.

(Step 2) Allocate $r$ to every TS child who completed fast prefetching (the child’s latest video time stamp is the same as the parent’s latest video time stamp).

(Step 3) Compute $\tilde{r} = U/n$, $\tilde{r} \geq r$. $U$ is the remaining uplink bandwidth after Step 2, and $n$ is the number of TS children who have not been assigned bandwidth in the previous steps. If some TS children have a smaller downlink capacity than $\tilde{r}$, the bandwidth allocated to them is curtailed in order not to overwhelm them. The remaining bandwidth is evenly allocated to the remaining TS children.

The algorithm is illustrated in Fig. 4.9. Note that the server performs no bandwidth partitioning because it supports no fast prefetching.
CHAPTER 4. TIME-SHIFTED STREAMING

4.4 Experimental Results

In the ns-2 experiments, 150 peers join and leave repeatedly with an average lifetime of 120 seconds. We used the *Mother and Daughter* sequence, encoded at $R = 420$ kbps, using H.264/AVC with an intraframe period of 15 frames. Two B frames were encoded between anchor frames. The video server’s uplink bandwidth was set to $20R$. $2R$ was allocated to LS peers, and the rest of the server’s uplink bandwidth was allocated to TS peers. The peer uplink bandwidth was set to $2R$ and the download bandwidth was set to $4R$. On joining at time $t$, half of the peers watched the live stream. The rest of them watched the stream shifted by a random delay ranging from 0 to $t$.

First, we examine the system without fast prefetching. Fig. 4.10 is the plot of the experimental results for video availability among peers. The value associated with the pair $(t, x)$ represents the average number of peers that possess video at position $x$ at time $t$. The result closely matches the video availability predicted by the model in Section 4.2 and shown in Fig. 4.4. Fig. 4.11 shows the video availability
for the model and the experiments in terms of the number of TS peers that possess video at position $x$ ($0 \leq x \leq 900$) at time 900s (no LS peers were present). It is notable that video availability peaks at 450s, the half point of the video available at 900s, under the assumption of random video access. As with Fig. 4.4 and 4.10, the video availability predicted by the model closely matches the averaged video availability of the experiments. As the session proceeds, only a small neighborhood of the video near the live stream is cached by the majority of the peers. The extent of the neighborhood is determined by several factors, one of which is the average peer lifetime. The expected number of peers that cache a past portion continues to decrease over time, as expected from the analysis.

![Figure 4.10: Video availability averaged over 100 simulation runs. ($\lambda = 1.125$ joins per second, $1/\mu = 120$ seconds, and $\alpha = 0.5$ are used.)](image)

Next, we study the system behavior when fast prefetching is employed. In Fig. 4.12, the average number of direct children of the server is plotted for three different cases:

- No fast prefetching is performed.
• Fast prefetching with parent selection minimum hops. For both TS and LS peers, parents that minimize the number of logical hops to the server are selected.

• Fast prefetching with parent selection maximum throughput. The parent selection algorithm described in Section 4.3.2 is used (\(D\) is set to 10 seconds).

The graphs show that with fast prefetching, fewer connections directly to the server are established because peers find their requested video among other peers more often than without fast prefetching. The proposed parent selection algorithm further reduces the server load, compared to the case where parent selection minimum hops was used.

As the session progresses, more TS peers connect to the server. This causes peers to experience less peer churn, leading to higher video quality at the cost of server bandwidth. Since only few TS peers need to connect to the video server at the early stage of a session, we evaluate video quality for the first 900 seconds. When there are missing or late packets that do not arrive in time, the last fully decoded frame is
displayed until the next I frame arrives. Table 4.1 summarizes average video quality and server load. The table indicates that fast prefetching achieves higher or similar video quality with reduced server load. We observe that fast prefetching with parent selection maximum throughput outperforms no fast prefetching by 2 dB for TS peers with about 40% sever load reduction. In Fig. 4.12, the server load remains low at the early stage of the session because TS peers can easily find peers to connect to. Toward the end of the session, in contrast, more TS peers need to receive video from the video server as fewer peers are found to cache video.

Figure 4.12: Comparison of server load over time (averaged over 10 simulation runs).

Finally, we investigate the sensitivity of fast prefetching against $D$. Recall that $D$ denotes a look-ahead time, the time a TS peer looks ahead to compute the expected download. In the experiment, we had 75 TS peers with uplink speed of $2R$, with 60 seconds of average lifetime and 6 seconds of average off-time. Simulations ran for 900 seconds and the average download speed at which a peer received video was averaged over 20 simulations. Fig. 4.13 shows the average download rate, which is normalized by the video bitrate. We observe that the average download peaks near
Table 4.1: Effects of fast prefetching (computed over the first 900 seconds). Video quality is represented by the average PSNR of peers. The average PSNR is computed at the end of the simulation by taking the average of the PSNR value of the video frames displayed at all peers. For missing frames, copy error concealment (display of the previously decoded frame) is used. Server load is the average number of direct children of the server. Hops and throughput are used as parent selection criteria.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Without prefetching</td>
<td>hops</td>
<td>40.6</td>
<td>40.4</td>
</tr>
<tr>
<td>With prefetching</td>
<td>hops</td>
<td>41.0</td>
<td>42.2</td>
</tr>
<tr>
<td></td>
<td>throughput</td>
<td>41.5</td>
<td>42.3</td>
</tr>
</tbody>
</table>

0 and decreases as $D$ increases, indicating that large $D$ may overestimate expected downloads because selected parents may depart the system. However, the download speed is not very sensitive to the value of $D$, showing only a 3% drop from the peak value for $1 \leq D \leq 20$. The download rate becomes nearly flat as $D$ approaches the average peer lifetime. It is worth noting that when $D \approx 0$, the maximum-throughput algorithm in Eq. 4.7 can be viewed as a maximum-download-rate algorithm; when $D \approx 0$ and $l_i$ is moderate, $\min(D \cdot q_i + l_i, D \cdot r_i)$ approximates to $D \cdot r_i$, thereby $\arg \max_i \min(D \cdot q_i + l_i, D \cdot r_i) \approx \arg \max_i r_i$.

### 4.5 Summary

In this chapter, we have demonstrated how SPPM can be extended to support time-shifted streaming without any fundamental changes in the system architecture. With fast prefetching, TS peers can mitigate video disruption when they are disconnected from the overlay due to peer churn. Fast prefetching also alleviates server load by expediting video dissemination among peers. Fast prefetching is further enhanced when it is combined with the maximum-throughput parent selection. We also proposed a
model to estimate the number of peers that possess the required video segments cached in their buffers. The model and the experiments revealed that few peers request time-shifted streams from the server at the early stage of the session because they can connect to peers who have been caching the live stream, and that the server load increases as the session progresses since a smaller number of peers are likely to have the requested time-shifted streams.

Figure 4.13: Normalized average download rate with respect to $D$, look-ahead time.
Chapter 5

Mobile Peer-to-Peer Streaming

In Section 3.1.1, we pointed out that the heterogeneity of the uplink bandwidths of peers has influenced the design of SPPM. The heterogeneity in the uplink bandwidth arose mostly due to the access networks that the peers connect to, and peers were assumed to be identical in the way they consume video content. In this chapter, the heterogeneity inherent in peers themselves, such as downlink speed, display size, and multimedia capability, is taken into account. Mobile devices, in particular, are expediting the growth of the heterogeneity. In the last decade, mobile devices such as smart phones and Personal Digital Assistants (PDAs) have become ubiquitous. However, live streaming to mobile devices is still a challenging task due to their heterogeneity. In typical streaming systems, live media adaptation is performed to meet the requirements of heterogeneous mobile users. For video, media adaptation is often achieved by video transcoding [187, 67, 25]. For mobile devices, video transcoding converts an original video bitstream to a new bitstream for a different encoding standard, smaller spatial resolution, reduced frame rate, or reduced quality (due to coarser quantization). However, transcoding poses a considerable computational burden on the streaming server because mobile devices often require individually customized transcoding.

In this chapter, we extend SPPM to incorporate mobile peers. Henceforth, we refer to personal computers or set-top boxes as fixed nodes, as opposed to mobile nodes. By harnessing the processing power of the fixed nodes, the transcoding burden of
the servers can be reduced or eliminated. On the other hand, due to their limited resources (battery, uplink speed), mobile nodes in SPPM are treated as leeches, i.e., peers that only receive packets but do not relay the packets to other peers. Moreover, videos customized for individual mobile devices make relaying video less appealing. To address adversarial effects due to peer churn, we propose a novel distributed transcoding scheme that allows more than one fixed node to perform transcoding for a mobile node. Even if a mobile device loses some of its parents, it may still receive substreams from the other parents. Then the proposed transcoding scheme allows the mobile device to decode the incoming video partially, thereby achieving graceful video degradation. In addition, the proposed transcoding scheme distributes transcoding overhead to multiple fixed nodes. We implement the proposed distributed transcoding scheme in a way that conforms to the H.264/AVC baseline profile. This allows any H.264/AVC decoder to play back the video produced by the proposed scheme.

The chapter is structured as follows. Section 5.1 describes the proposed distributed transcoding and its implementation. Section 5.2 presents the extension to the existing SPPM protocol. In Section 5.3, we analyze the effect of peer churn on distributed transcoding. In Section 5.4, we provide simulation results.

5.1 Distributed Transcoding

In this section, we describe how video transcoding is performed at one or multiple locations for a mobile node in our P2P system. We consider a P2P system that consists of fixed nodes and mobile nodes; fixed nodes are peers that receive and consume the original video emanating from the video source. Mobile nodes are peers that cannot receive the original video due to limited downlink bandwidth, or/and cannot consume the original video due to limited video decoding capabilities. Thus, it is desirable to perform video transcoding to adapt the original video to mobile nodes. Fixed nodes perform transcoding to adapt the original video according to the individual requirements of each mobile node. In this study, we adopt the cascaded transcoding scheme shown in Fig. 5.1.
5.1.1 Interleaved Distributed Encoding

The multiple parent approach is a popular solution to provide robustness to peer churn [40, 138, 163]. In this approach, a peer has multiple peers as parents. When a parent disappears, only a subset of video packets are lost, which allows for graceful video degradation. We propose a distributed transcoding scheme that allows multiple fixed nodes to perform transcoding for a mobile device. A mobile node selects multiple fixed nodes as parents and the parents perform transcoding collaboratively. A straightforward scheme would be that each parent transcodes the entire original video and delivers a disjoint substream of it to a mobile node. When one of its parents disconnects, the mobile node asks for retransmissions of the missing substream from other parents. However, such a straightforward scheme needlessly wastes computing power at the parents.

To reduce processing redundancy, yet achieve robustness with multiple parents, we propose interleaved distributed transcoding (IDT), which is illustrated in Fig. 5.2. In the illustration, $K = 4$ parents are generating transcoded substreams with a Group of Pictures (GOP) of 12 frames (the original GOP size is assumed to be larger than 14 in Fig. 5.2). This illustration demonstrates that the GOP size of the transcoded stream can be selected independently of the one of the original stream. In transcoding, the original video frames are first decoded. In this illustration, the decoded bitstream is downsampled to smaller frames in the spatial domain. The first frame in a GOP is coded as an I frame, and each following frame is coded as a P frame predicted from
CHAPTER 5. MOBILE PEER-TO-PEER STREAMING

Figure 5.2: Interleaved distributed transcoding: an example of 4 parents (IDT encoders) with a GOP of 12 frames. The original video stream is divided into 4 sub-streams. The first frame of a GOP is encoded as an I frame by all parents. To avoid duplicate transmission, only Parent 1 transmits I frames. The I frames that are not transmitted are depicted with dotted boundaries. The solid arrows represent coding dependency. The dotted arrows depict the frame copy operation. The original stream is downsampled before encoding.

The frame immediately preceding it in the substream. Parent \( i \) codes Substream \( i \), which includes Frame \( i, K + i, 2K + i, \ldots \), and each parent transmits every \( K^{th} \) frame in a disjoint manner. The I frames are encoded and used in prediction by all parents, yet transmitted by only Parent 1 to avoid duplicate transmission. Note that B frames can be employed within each substream to achieve higher coding gain.

The proposed distributed transcoding scheme achieves robustness against peer churn and distributes transcoding workload among multiple fixed nodes. The incurred cost is the redundancy in the transcoding bitstream due to lower temporal correlation between video frames, which will be examined in Section 5.4 in detail.
5.1.2 Decoding Transcoded Video

Substreams generated by multiple parents are transmitted to the destination mobile node. In Fig. 5.3, the substreams are assembled by the mobile node; it interleaves frames according to their positions in the GOP structure. As the substreams are interleaved, the frame copy bits contained in the other substreams are discarded for frames that are successfully received. The assembled bitstream is then passed to the decoder for playback.

Figure 5.3: Decoding at a mobile node. In this example, one of 4 substreams, Substream 3, is assume to be missing due to parent loss. The frame copy bits from Substream 2 are used in constructing the assembled bitstream.

When a parent disconnects, the corresponding substream becomes unavailable at the mobile node. If Parent 1 disconnects, then the mobile node requests missing I frames from one of the other parents (recall that I frames are encoded at every parent). For the frames of the missing substream, the frame copy bits from the available substream preceding the missing substream are used as a replacement. In Fig. 5.3, Parent 3 disconnects and Substream 3 becomes unavailable. To replace Frames 3, 7, 11, 15 in the missing substream, the frame copy bits in Substream 2 are
used. Note that the redundancy of frame copy bits in multiple substreams allows the assembled bitstream to be correctly played back even when more than one bitstream is missing. Since the assembly of substreams and the selective insertion of frame copy bits are performed at the bitstream level, we can avoid any modification to the decoder.

5.1.3 Implementing IDT

We implement the interleaved distributed transcoding (IDT) scheme in such a way that no decoder modification is required. The IDT generates no B frames and utilizes multiple reference frames for encoding P frames. This ensures that any decoder conforming to the H.264/AVC baseline profile can decode transcoded bitstreams. We refer to the H.264/AVC encoder that implements the proposed interleaved distributed transcoding as the *IDT encoder*. Suppose that *K* parents are involved in transcoding. The IDT encoders at the parents encode the first frame in a GOP as an I frame, which is identical across all the encoders. The remaining frames in a GOP are encoded as P frames. To encode Frame *n* as a P frame, Frame *n* − *K*, the previously encoded frame in the same substream, is used as a reference frame for motion-compensated prediction. Therefore, the IDT encoder is required to store *K* previously encoded frames. For this, we take advantage of the *multiple reference picture motion compensation* specified in the H.264/AVC baseline profile. It allows the short-term reference picture buffer to hold multiple reference pictures, in our case, *K* previously encoded frames.

We also employ the *reference picture reordering* specified in the H.264/AVC baseline profile to ensure the correct frames are used as a reference picture for motion prediction. The H.264/AVC standard provides the *P-SKIP mode*. For this coding type, neither a quantized prediction error signal, nor a motion vector or reference index parameter is transmitted [157]. The reconstructed signal is obtained similar to the prediction signal of a P-16x16 macroblock type that references the picture which is located at index 0 in the reference picture buffer. The motion vector used for reconstructing the P-SKIP macroblock is similar to the motion vector predictor (MVp)
for the 16x16 block, where MVp is computed as the median of the motion vectors of
the macroblock partitions or sub-partitions immediately above, diagonally above and
to the right, and immediately left of the current partition or sub-partition [92]. Since
the \textit{P-Skip mode} always refers to the frame located at index 0 in the buffer, we move
the previous frame in a substream to the index 0 location by the reference picture
reordering. As a result, the encoder refers to only the frame at the same location for
prediction, although there may be up to \( K \) pictures available in the buffer.

When the IDT encoder encodes every \( K^{th} \) frame, the remaining frames are en-
coded as an exact copy of the previously encoded frame. In Fig. 5.2, the IDT encoder
at Parent 1 encodes Frames \( P_2, P_3 \) and \( P_4 \) as a copy of Frame 1. \textit{Frame copy} encodes
frames with negligible computational complexity at the cost of about 1 \( \sim 2 \) % of
control bits\footnote{The length of the control bits is 6 \( \sim 7 \) Bytes for a QCIF-resolution video. The control bits are
about 4 kbps for 4 parents, GOP size = 12 and 30 fps.} added to the transcoded video. \textit{Frame copy} not only avoids encoding
unnecessary frames, but also embeds control bits for error concealment in the bit-
stream. This allows error concealment to be done at the bitstream level and ensures
that any standard-compliant decoders can decode the assembled bitstream without
having to detect a parent or packet loss by itself.

\section{Protocol Extension}

In order for the mobile peer to join the SPPM system, it communicates with the server
and parent candidates selected by the server. Once it is connected to the system, it
continually exchanges control messages to continue to receive video packets. Details
can be found in Appendix A.2.

\textbf{Communications between a Mobile Node and the Server}

A mobile peer contacts the server to ask for a partial list of the fixed peers in the
system. It sends the following information to the server: device class, display size,
video codecs, and maximum download bitrate. The server replies with information
 Communications between a Mobile Node and Parent Nodes

- **Probing:** A mobile peer collects information from parent candidates, such as available bandwidth, the depth in the overlay tree of the SPPM network, frame ID of the recently decoded frame, etc.

- **Attaching:** The mobile peer selects several parent candidates to connect to. The criteria for parent selection may be determined by various factors (e.g., parents physically closest to the mobile node, or parents that are serving the minimum number of mobile nodes). The mobile peer connects to the selected parent candidates. As an active coordinator, the mobile peer determines the substream ID for each parent. Note that a parent has no need to communicate directly with other parents.

- **Connection maintenance:** The mobile node and its parents continue to exchange messages for status report or change in settings. Parent disconnection or mobile node disconnection is handled by the protocol.

### 5.3 Peer Churn Analysis

In this section, we discuss how distributed transcoding mitigates the effect of peer churn. For simplicity, we assume that a parent node’s lifetime is exponentially distributed, with an average of $1/\mu$ seconds. We further assume that a missing parent node can be replaced with another node in a recovery time that is also exponentially distributed, with an average of $1/\lambda$ seconds. Each parent node can fail independently from any other parent, and the mobile node’s recovery process of finding another parent is independent from the state of all other parents. We consider a Markov chain, where a state represents the number of live parents [105]. Fig. 5.4 illustrates a state-transition-rate diagram, where $K$ represents the total number of parents.
Figure 5.4: State-transition-rate diagram for a mobile node with $K$ parents. $1/\lambda$ is the average recovery time. $1/\mu$ is the average parent lifetime.

When a parent arrives, the system state jumps from State $i$ to $i + 1$. For a transition from State $i$ to $i + 1$ ($0 \leq i \leq K - 1$), $K - i$ concurrent recovery processes are performed; the transition rate is therefore $(K - i)\lambda$. When a parent leaves, the system state jumps from State $i$ to $i - 1$. For a transition from State $i$ to $i - 1$ ($1 \leq i \leq K$), $i$ parents are alive and have the same failure rate of $\mu$; the transition rate is therefore $i\mu$. We can now compute the stationary state probability distribution of this Markov chain by using the following relationship:

$$\lambda \pi_{(K-1)} = K \mu \pi_K$$  \hspace{1cm} (5.1)

$$\sum_{i=0}^{K} \pi_i = 1,$$  \hspace{1cm} (5.2)

where probability flow conservation and probability conservation apply. By expressing $\pi_k$ ($k \neq 0$) with $\pi_0$, we obtain

$$\pi_0 = \left\{ \sum_{i=0}^{K} \binom{K}{i} \left( \frac{\lambda}{\mu} \right)^i \right\}^{-1}$$

$$= \left( \frac{\lambda}{\mu} + 1 \right)^{-K}.$$  \hspace{1cm} (5.3)

$$\pi_i = \left( \frac{\lambda}{\mu} + 1 \right)^{-K} \binom{K}{i} \left( \frac{\lambda}{\mu} \right)^i, \hspace{0.5cm} 0 \leq i \leq K.$$  \hspace{1cm} (5.4)
Note that the binomial theorem is applied in Eq. 5.3. We can relate the states depicted in Fig. 5.4 with the effective frame rate of the decoded video. Suppose that \( f \) is the frame rate of the transcoded video. When the mobile node is in State \( K - 1 \), then one substream is missing. With copy error concealment, previously decoded frames are displayed in lieu of the frames of the missing substream. Thus the effective frame rate becomes \( (\frac{K-1}{K})f \) (no packet loss over the wireless channel is assumed for this discussion). For instance, when \( K = 2 \), if the mobile node is in State 1, then the effective frame rate is \( 0.5f \). When \( K = 4 \), then if the mobile node is in State 3, the effective frame rate is \( 0.75f \).

Table 5.1 shows the average fraction of time during which a particular number of parents (substreams) of a mobile node are missing under different peer churn rates. \( \lambda/\mu \), denoted by \( \alpha \), indicates the ratio of the parent average lifetime to the parent average recovery time. As \( \alpha \) increases, the state probability of State \( K \) (no parents are missing) also increases and the state probability of all the other states decreases. This is obvious because the mobile node is more likely to have all parents alive when the recovery time is shorter or the average lifetime is longer.

In Table 5.2, the effect of peer churn on the number of missing parents is shown with the total number of parents \( K \) varied from 0 to 4. As a mobile node connects to more parents, the probability that all parents are missing approaches 0, whereas the probability that at least one parent is missing increases (when \( \alpha \) is large, the probability of missing one parent (State \( K - 1 \)) increases linearly as \( K \) increases). Note that regardless of \( K \), the average fraction of missing frames is a constant \( \frac{1}{\alpha+1} \); in State \( i \), \( \frac{K-i}{K} \) fraction of packets are lost due to \( K - i \) missing parents. Then, \( \Psi(K) \), the average fraction of packets lost given \( K \) parents is expressed as

\[
\Psi(K) = \frac{1}{\alpha+1}
\]

(5.5)
CHAPTER 5. MOBILE PEER-TO-PEER STREAMING

This result shows that connecting to multiple parents does not affect the average number of missing packets. Instead, it alleviates the degree of video degradation at the expense of more frequent video degradation\(^2\) [115].

Table 5.1: The average fraction of time during which a particular number of parents of a mobile node is missing. The total number of parents is set to 4. \(\lambda/\mu\) is the ratio of the average parent lifetime to the average recovery time.

<table>
<thead>
<tr>
<th>(\lambda/\mu)</th>
<th>Number of missing parents</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.683 0.273 0.041 0.003 0</td>
</tr>
<tr>
<td>30</td>
<td>0.877 0.117 0.006 0 0</td>
</tr>
<tr>
<td>50</td>
<td>0.924 0.074 0.002 0 0</td>
</tr>
</tbody>
</table>

Table 5.2: The average fraction of time during which a particular number of parents is missing given a different total number of parents (\(\alpha\) is set to 30).

<table>
<thead>
<tr>
<th>Total number of parents</th>
<th>Number of missing parents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.968 0.032 N/A N/A N/A</td>
</tr>
<tr>
<td>2</td>
<td>0.936 0.062 0.002 N/A N/A</td>
</tr>
<tr>
<td>3</td>
<td>0.906 0.091 0.003 0 N/A</td>
</tr>
<tr>
<td>4</td>
<td>0.877 0.117 0.006 0 0</td>
</tr>
</tbody>
</table>

5.4 Experimental Results

We conducted computer simulations to evaluate the proposed distributed transcoding scheme in a P2P environment. We implemented the IDT algorithm by modifying the x264 encoder [180]. The x264 encoder is an open source library for encoding videos in the H.264/AVC syntax. Six video sequences in CIF resolution were used as the original videos, encoded by using the H.264/AVC main profile. The GOP size was 15

\(^2\)We assumed no packet loss in the analysis. In reality, packet loss may be affected by video bitrate, which is subject to \(K\).
frames, and two B frames were generated between the anchor frames. The frame rate of the original video was 30 fps. The original video was streamed live to a population of SPPM peers from the video source. Peers incrementally constructed the overlay as they joined the system [163]. When a fixed node was connected to the system, it received the original video as long as its downlink capacity was higher than the video bitrate. For simplicity, we assumed that all fixed nodes had downlink capacity larger than the video bitrate.

When a mobile user joins the system, it searches for \( K \) fixed nodes that have available uplink bandwidth and processing power. We assumed that the number of fixed nodes exceeded \( K \). After the mobile user finds \( K \) fixed nodes as parents, it assigns them unique Parent IDs (from 1 to \( K \)). Then it requests them to transcode disjoint sets of video frames (substreams). For the synchronization of substreams, parents add meta data to substreams, such as the time stamp of a GOP. During the parent-coordination process, the mobile node examines its device-specific profile, such as the media decoding capability, display size, and user’s preference. It also detects time-varying parameters including the remaining battery capacity and the maximum downlink bandwidth of the wireless channel. Based on the collected information, the mobile node determines the video quality (e.g., quantization parameter), frame rate, and spatial resolution. In our simulations, we kept the frame rate of the original video. Spatial resolution was reduced from CIF to QCIF. For a transcoded video, the GOP size was set to 24. The quantization parameter and the number of parents were varied across different simulations.

The fixed node’s lifetime is exponentially distributed with an average of 90 seconds. When a fixed node serving as a parent leaves the system, its child node finds a different fixed node to recover the missing substream. The recovery time is exponentially distributed with an average of 3 seconds. Although the IDT provides error resilience on its own, it is sensitive to I frame loss. When Parent 1 failure is detected, the mobile node selects one of its available parents as the new Parent 1. When I frames are lost due to a lossy channel, retransmission is requested for the missing I frames. To avoid self-congestion, retransmissions of P frames are not requested.
Figure 5.5: Rate-distortion curves with and without packet loss for conventional transcoding (1 parent) and interleaved distributed transcoding (2 and 4 parents). (a) Foreman sequence; (b) Mother & Daughter sequence; (c) Container sequence; (d) Mobile sequence; (e) News sequence; (f) Paris sequence.
Fig. 5.5 shows the simulation results for single-parent transcoding and the interleaved distributed transcoding (two and four parents). A distortion measured in PSNR is computed between the input stream to the IDT encoders (the bistream after the intermediate processing unit) and the assembled stream. The solid curves show the rate-distortion performance without packet loss and without peer churn, in a case called “no-packet-loss”. The single-parent shows the highest coding efficiency in this case because temporal redundancy is removed most effectively. When more parents are involved with transcoding, the distance between frames in a substream increases and this lowers inter-frame temporal correlation.

Now we consider the lossy scenario, where video degradation is caused by two different sources: packet loss over the wireless channel and peer churn. The wireless channel is modeled by the Gilbert-Elliot model, as a discrete-time Markov chain with two states. Table 5.3 shows the Gilbert model transition probabilities based on the GSM network, presented in [106]. During the good state there is no packet loss, whereas during the bad state the channel produces packet loss with probability 0.5. State transitions occur according to Table 5.3 at the transmission of each packet. For peer churn, a burst of packets that belongs to a substream is lost when the corresponding parent leaves. The dashed curves in Fig. 5.5 indicate the degraded video quality in the lossy scenario.

Table 5.3: Probabilities used in the Gilbert-Elliot model.

| State  | Pr(i)   | Pr(1|i) | Pr(0|i) |
|--------|---------|---------|---------|
| 0 (Good) | 0.9449  | 0.0087  | 0.9913  |
| 1 (Bad)  | 0.0551  | 0.8509  | 0.1491  |

Some sequences, such as Foreman sequence, contain more motion than other sequences, such as Mother & Daughter, and the concealment of lost frames is often more difficult. Fig. 5.5 also shows that the single-parent case suffers from a significant quality drop from the no-packet-loss case. This indicates a huge fluctuation in video quality occurs when the mobile node loses its only parent. On the contrary, distributed transcoding (both two and four parents) exhibit a lower quality degradation than single-parent, which implies a lower variance in video quality. We also
observed that the performance difference between two parents and four parents is negligible except for the Mobile sequence. Recall that two parents outperformed four parents in the no-packet-loss case. This shows that the variance of the video quality is smaller by more parents because the adverse impact of packet loss is alleviated with more substreams. This result is analogous to that from the analysis of peer churn. In Section 5.3, we showed that the impact of parent disconnect, in terms of the effective frame rate, is smaller with more parents at the cost of longer video degradation periods.

We compared the performance of IDT against Multiple Description Coding (MDC) [23]. The MDC in [23] is similar to our IDT scheme; however, each interleaved video frame sequence contains its own I frames, and thus each substream can be decoded independently. For a fair comparison with IDT, the GOP size of each substream is made identical to the GOP size used for IDT. This ensures the ratio of the number of I frames and the number of P frames in the bitstream is identical for both MDC and IDT. Fig. 5.6 shows the performances of MDC and IDT for the six video sequences. IDT consistently outperforms MDC for both the lossy scenario and the scenario without packet loss. A plausible explanation is that in MDC, a GOP in a substream spans a larger duration of the original video than a GOP in the assembled stream. For instance, for K=4 parents and a GOP of 24 frames, IDT encodes every 24\textsuperscript{th} frame as an I frame. In contrast, MDC encodes every 96\textsuperscript{th} frame as an I frame. Reducing the GOP size may alleviate this slow refresh, but it results in a worse rate-distortion performance due to the higher ratio of the number of I frames to the number of P frames.

Next, we measured the time spent by the modified x264 encoder. Since a fixed node performs decoding for its own playback, we do not include decoding time. We also ignored the intermediate processing time, which is much smaller than the encoding time. We averaged the encoding times from 100 experiments\textsuperscript{3} (Table 5.4). Encoding a frame of the Foreman sequence using single-parent transcoding required about 6.6 ms on average. When the frame rate of the transcoded bitstream is 30 fps,
Figure 5.6: Video quality comparison of IDT and MDC. (a) Foreman sequence; (b) Mother & Daughter sequence; (c) Container sequence; (d) Mobile sequence; (e) News sequence; (f) Paris sequence.
then about 20% of the CPU cycles are required for a mobile user. As more parents are involved in transcoding, each parent spends less time because the generation of frame copy bits has low complexity.

Table 5.4: Average CPU time required for encoding a frame at one of parents.

<table>
<thead>
<tr>
<th></th>
<th>1 parent</th>
<th>2 parents</th>
<th>3 parents</th>
<th>4 parents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>6.60 ms</td>
<td>4.14 ms</td>
<td>3.19 ms</td>
<td>2.74 ms</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>4.39 ms</td>
<td>2.86 ms</td>
<td>2.41 ms</td>
<td>2.14 ms</td>
</tr>
<tr>
<td>Container</td>
<td>4.38 ms</td>
<td>2.70 ms</td>
<td>2.20 ms</td>
<td>1.96 ms</td>
</tr>
<tr>
<td>Mobile</td>
<td>7.25 ms</td>
<td>4.03 ms</td>
<td>2.49 ms</td>
<td>2.42 ms</td>
</tr>
<tr>
<td>News</td>
<td>3.08 ms</td>
<td>2.54 ms</td>
<td>2.06 ms</td>
<td>1.82 ms</td>
</tr>
<tr>
<td>Paris</td>
<td>4.48 ms</td>
<td>2.77 ms</td>
<td>2.28 ms</td>
<td>1.98 ms</td>
</tr>
</tbody>
</table>

5.5 Summary

In this chapter, we provided an extension to SPPM in order to stream video to mobile devices in a heterogeneous peer-to-peer environment. To satisfy different requirements of various mobile devices, fixed peers in the P2P network contribute their processing power to perform video transcoding. The proposed interleaved distributed transcoding (IDT) is shown to reduce the amount of computation that each parent has to perform. The IDT scheme also provides error resilience against packet loss due to peer churn and to adverse wireless channel conditions. By analyzing the effects of peer churn, we showed that distributed transcoding balances the degree of video degradation with the occurrence of video degradation. Simulation results show that IDT outperforms multiple description coding (MDC) by 1 to 1.5 dB. We also implemented a real-time IDT encoder and verified the results of our analysis and simulations.
Chapter 6

Conclusions

In this dissertation, we address the problem of one-to-many video distribution by proposing Stanford Peer-to-Peer Multicast (SPPM), a low-latency and robust peer-to-peer (P2P) video streaming protocol. SPPM achieves cost-effective streaming by harnessing the resources of participating user devices, called peers, thereby reducing or eliminating the need for expensive dedicated servers. SPPM achieves low-latency and robust streaming by explicitly constructing multiple complementary trees among peers in a decentralized manner. In the following section, we summarize the lessons we learned from the system design, analysis, simulations, and deployment of SPPM. Finally, several open problems are proposed to attract more researchers for the study of P2P streaming.

6.1 Lessons Learned

The multicast trees in SPPM are built by adding peers to the overlay in the order peers arrive in the system. Since the trees grow incrementally as peers join the system, peers’ join sequences have a strong impact on the system performance. In particular, system clogging exists in a system of peers with heterogenous uplink bandwidths. Clogging occurs when leeches, peers with uplink bandwidth lower than the video bitrate, block all data paths. Experiments and analysis found that clogging is more likely as the peer population becomes bigger or the server bandwidth is smaller. Using
the un-leeching protocol, SPPM overcomes clogging by rearranging the position of the peers in the overlay. Like the un-leeching protocol, the tree-height reduction protocol is a part of the active overlay management that allows peers in the SPPM system to proactively rearrange peer positions. As the multicast trees are used as a data dissemination structure, the trees are built and maintained in a way that they achieve lower end-to-end delays and less video disruptions. By simulations and a mathematical model, a trade-off was shown to exist between the end-to-end delay and the average tree height (number of logical hops), suggesting that a moderate number of trees (4 to 8) can balance them. In SPPM, peers allocate their uplink bandwidths to peer types. Three peer types are defined according to the presence and type of NATs/firewalls. It is shown that the proposed bandwidth allocation can maximize the self-sustainability with a fixed parameter set when the system is sufficiently over-provisioned.

Next, a tree-based P2P live streaming system like SPPM is shown to support time-shifted streaming without any fundamental changes in the system architecture. Time-shifted streaming allows users to control the play-back of video. With simulations and mathematical analysis, it is shown that time-shifted streaming may increase requirements for server bandwidth over time. Fast prefetching alleviates the requirement of server bandwidth by expediting video dissemination among peers. Fast prefetching is possible because the relaxed transfer deadline of time-shifted streams and peers’ extra uplink bandwidths are exploited. In addition, fast prefetching mitigates video disruption when peers are disconnected from the overlay due to peer churn. The video availability improves more significantly when fast prefetching is combined with the maximum throughput parent selection. This shows that the overlay may look very different depending on what objectives are pursued in the overlay construction.

Finally, SPPM was extended to incorporate heterogeneous peers, in particular, mobile devices. Since mobile devices are often limited in their display size, network speed, and video play-back capability, video transcoding is required for adapting video. The proposed transcoding algorithm, interleaved distributed transcoding (IDT), utilizes the computational powers (e.g., CPUs) of fixed peers (e.g., personal computers) to perform transcoding. IDT reduces the amount of computation that
each fixed peer has to perform for a mobile peer. At the same time, IDT improves error resilience against packet loss due to peer churn and to adverse wireless channel conditions. By analyzing the effects of distributed transcoding on peer churn, it is shown that distributed transcoding does not reduce the average number of missing packets. Instead, it balances the degree of video degradation with the occurrence of video degradation.

### 6.2 Open Problems

Our research gives rise to many interesting research problems. In Section 3.4, we discussed the optimal number of complementary trees. Although we suggested to use a moderate number of trees, we also see that the optimum number changes over time with respect to the peers participating in a session. Therefore, dynamic adaptation of the number of trees and evaluating its benefits and control traffic is an interesting topic to study. For simulations in this dissertation, we focused on a large population of users. In practice, however, many video sessions are small, e.g., attracting only a few concurrent users. Thus our study leaves room for further research on the performance of a P2P system with a small number of users. We have discussed how to un-leech the overlay in case the system suffers from clogging, which occurs when the system is unable to achieve self-sustainability. Another approach to address clogging is to dynamically adapt the video bitrate at the video encoder, so that the average uplink bandwidth contributed by peers does not exceed the video bitrate. The average uplink bandwidth can be measured by unbiased distributed sampling [173], or by a central server.

In Chapter 4, time-shifted streaming was studied to support change in playback position in time. Although we showed the server bandwidth behavior over time, further study is desirable. We proposed the maximum-throughput parent selection for TS peers. However, one might also use different parent selections that strive to optimize other metrics, such as video disruption experienced by peers. For caching video, we considered the continuous bitstream consisting of past video packets stored in peers’ local storage. One can devise more sophisticated caching algorithms, which
may allow multiple chunks of video that are not necessarily continuous in bitstream. Server bandwidth allocation is also an important research problem. We have found that time-shifted streaming often requires more server bandwidth than live streaming because TS peers that request past portion of video need to connect to the server if the requested video chunk is not available from other peers. Therefore, the server needs to divide its uplink bandwidth between live streaming and time-shifted streaming dynamically in order to maximize the overall video quality of peers.

Finally, much interesting research may be carried out to extend our study on mobile peer support in a P2P system. For instance, IDT can be enhanced with adaptive algorithms: one can dynamically change the number of parents in conjunction with the selection of the optimal number of parents given the condition of a wireless channel and the requirements of a mobile peer. Rate control, such as temporary disabling of some parents or video frame skipping, may be worth studying as well. In this dissertation, we treated mobile peers individually. However, clustering mobile peers into a limited number of profile groups will improve system scalability in some scenarios; for instance, where a large population of mobile peers is present. Along the same line, estimation of the maximum number of mobile peers a P2P system can accommodate is also crucial in the evaluation of scalability. This may require the quantification of CPU usage and transcoding load. Although virtually no SVC decoders for mobile devices are available at the writing of this dissertation, this may change in the near future. When SVC becomes prevalent, it will be interesting to compare on-demand transcoding, such as IDT, with SVC-based streaming in the P2P context.
Appendix A

SPPM System Extensions

In Section A.1, the un-leeching protocol and child promotion protocol are presented. Section A.2 presents the architecture of the SPPM fixed peer and the SPPM mobile peer, and the protocol associated with the mobile extension. The descriptions of the basic SPPM protocol can be found in [162].

A.1 Active Overlay Management

A.1.1 Un-leeching Protocol

Suppose that Peer C contacts the source S to join the system. The source S responds with a list of randomly drawn peers that might potentially serve as parents. By probing these peers, Peer C collects partial information about the P2P overlay. If Peer C finds a peer with available bandwidth, it connects to the peer. When Peer C fails to find any peer with available bandwidth, it concludes that the system is clogged. Then, Peer C performs self-admission control by comparing its own uplink bandwidth, $R_C$, with $R_{\text{thres}}$. A threshold $R_{\text{thres}}$, set equal to $R$, is used to determine whether an incoming peer is a leech. When $R_C$ is below $R_{\text{thres}}$, it classifies itself as a leech and is not allowed to join the system. If $R_C$ exceeds $R_{\text{thres}}$, then Peer C selects the leech with the shortest path to the source among those that have responded to the probing.
Fig. A.1 illustrates the communication between the peers involved in the un-leeching procedure. Let Peer $B$ denote the selected leech. Peer $C$ sends “Swap Request” to Peer $B$’s parent, Peer $A$, in order to request peer swapping between Peer $B$ and $C$. Peer $A$ accepts Peer $C$ as a new child and disconnects Peer $B$. Peer $A$ then replies to Peer $C$. When Peer $C$ receives the “Reply to Swap Request”, it sends “Swap Notification” to Peer $B$. Once the swapping is completed, Peer $A$ becomes Peer $C$’s parent and Peer $C$ becomes the new parent of Peer $B$. The lock at Peer $C$ is released when the reply from Peer $A$ arrives. When Peer $A$ sends no reply, then the lock expires and Peer $C$ repeats the un-leeching process with another leech.

If no leeches are found during the probe step, Peer $C$ repeats the procedure by obtaining a new list of potential parents from the source $S$ until it finds a leech to swap with.

![Diagram of the un-leeching procedure](#)

Figure A.1: The un-leeching procedure. Peer $C$ requests peer swapping with Peer $B$ (leech) to Peer $A$ (leech’s parent).
A.1.2 Child Promotion Protocol

Suppose that Peer $X$ initiates child promotion when one of its child nodes reports higher out-degrees than its own out-degree. If there is more than one child node that meets this triggering condition, Peer $X$ chooses the child node with the largest out-degrees. If there is a tie, the child with the largest uplink bandwidth is chosen. Suppose that Peer $Y$ is Peer $X$’s child with the largest out-degrees, which is higher than Peer $X$’s out-degree. If Peer $Y$ has no more available bandwidth, Peer $Y$ selects one of its current child nodes, Peer $Z$. Otherwise, Peer $Z$ is not selected and is henceforth ignored. Peer $Z$ connects to Peer $X$ as a child by taking over the bandwidth previously occupied by Peer $Y$. The rest of the child nodes and their descendant nodes are kept intact.

Fig. A.2 illustrates the communication between the peers involved with the child promotion procedure. As in the peer swapping for un-leeching the overlay, the peers that change their parent nodes, Peer $X$, $Y$, and $Z$, enter the lock state until the swap is complete. Peers in the lock state prevent themselves from being involved with other concurrent overlay rearrangement.

![Diagram of child promotion procedure]

Figure A.2: The child promotion procedure. Peer $X$ requests swapping with Peer $Y$. Peer $Z$ attaches to Peer $X$ when the swap is complete.
The lock state expires when no reply arrives within a deadline, which allows sufficient time for control packets to travel among the peers. Since peer swapping involves two or three closed-loop communications (one between Peer X and Y, another between Peer Y and G, and the other (if any) between Peer Z and X), peer swapping is completed depending on how far the peers are physically distant from each other\(^1\).

### A.2 Mobile Peer-to-Peer Streaming

#### A.2.1 SPPM Fixed Peer

As a fixed peer, the SPPM peer is extended to include a mobile child manager, a decoder, an intermediate processing unit, an encoder, a packetizer, and a queue for transcoded video, as shown in Fig. A.3.

---

\(^1\)In ns-2 simulations, the average time for peer swapping was around 200 ms, which was 2.5 times the largest round-trip time between peers in the network used for the experiments.
• The mobile child manager creates, maintains, and terminates a service for a mobile peer. When a new mobile peer requests a connection, the manager creates a dedicated encoder.

• The decoder is used to decompress the original video stream. The decompressed video is stored as a form of a raw frame, such as a video signal in the YUV format in a buffer, called Decoded Raw Video Buffer. Due to packet loss or internal errors in decoding, some raw frames may be missing in the buffer. Each frame in the buffer is tagged with both the global Group-Of-Picture (GOP) ID and local frame ID. These meta data are piggy-bag in the video packets and referred to by the interleaver at the mobile peer.

• The intermediate processing unit is used for modifying raw frames before they are passed into the encoder. Resizing (e.g. down-sampling or changing the ratio of the number horizontal and vertical pixels), cropping, and/or frame rate reduction are performed at this unit. For each mobile peer, a different intermediate processing may be performed.

• The encoder encodes the modified raw video signals. For each mobile peer, a different set of encoding parameter may be used, such as the GOP size, encoding standard (e.g., H.263 or H.264/AVC), and the total number of parents.

• The packetizer generates video packets for network transmission. It also adds meta data to packets, such as packet ID, GOP ID, substream ID, and a timestamp.

• The queue for transcoded video is used to store video packets. Packets in the queue are passed to the NIC (Network Interface Card) for transmission.

A.2.2 SPPM Mobile Peer

The mobile peer consists of the input/output NIC, a control unit, a video manager, and an interleaver. A control unit processes incoming control messages and generates
response messages or new control messages to the clients parents. The component diagram is shown in Fig. A.4.

![System Architecture of the SPPM Mobile Peer](image)

Figure A.4: System architecture of the SPPM mobile peer.

- The video substream manager requests the next video packets and processes incoming video packets. Each substream is a bitstream of an H.264/AVC raw video signal. The video manager divides a substream into frames by detecting their boundaries (e.g., NAL units). The manager also marks each frame with relevant meta data such as POC (Picture Order Count), GOP ID, Frame ID, and the substream ID. The video manager requests retransmission of missing I frames. It also detects parent disconnect or evaluates the download channel status and triggers necessary actions at the control unit.

- The interleaver assembles the frames of the substreams into a single output bitstream. When there is a missing frame due to a missing packet or missing substream, corresponding copy frame control bits are substituted in order to conceal missing frames. The interleaver discards old frames or unnecessary frames in the queues for the substreams. The interleaver starts assembling after
a time-out. The time-out interval can be adjusted so that the initial buffering
time and the packet reception ratio (excluding late or missing packets) are balanced.

- The control signal processor processes incoming control messages. It also trig-
gers events, such as coordinating parents or detecting a missing parent

A.2.3 Protocol Extension

When a mobile peer connects to the SPPM system, it communicates with the video
source and parent candidates (SPPM fixed peers). Once it connects to the system,
it constantly exchanges control messages to maintain the parents and receive video
packets.

Communications between the mobile peer and the tracker

A mobile peer contacts the tracker to obtain a partial list of fixed peers. It sends the
following information to the tracker: device class, device capabilities (display size,
codecs), maximum download bitrate. The tracker replies with information, such as
the bitrate of the original video and peer list.

Communications between the mobile peer and fixed peers

- **Probing:** The mobile peer collects information from parent candidates such
  as the available bandwidth, depth in the overlay tree, frame ID of the recently
decoded frame.

- **Attaching:** The client selects several parent candidates to connect to.
  - The mobile peer connects to selected parent candidates
  - As an active coordinator, the mobile peer determines the substream ID
    for each parent. Note that parents have no need to directly communicate
    with other parents.
– In the attach request message, the client specifies the following fields: Display size, down-sampling ratio, total number of parents, Parent ID:1 to $M$ ($M$ is the total number of parents), video quality parameter QP, GOP size, synchronization frame ID (Global frame ID used to assemble substreams at the mobile peer).

• **Connection maintenance**

– The client and its parents continue to exchange messages for status report or change in settings.

– When a parent failure is detected by noticing no video reception, the mobile peer triggers the rejoin procedure. If Parent 1, who is responsible for I-frames, fails, the mobile peer selects one of the other parents as Parent 1. The rest of the rejoin procedure is similar to the rejoin procedure of the basic SPPM protocol.

– When a mobile peer failure is detected by noticing no *Hello* messages, the fixed peer discards the mobile peers information and terminates the dedicated encoder.
Appendix B

Analysis of System Clogging

In Section 3.3, we defined a *leech* as a peer that does not contribute to a system at all, or contributes uplink bandwidth less than $R$, but that nevertheless receives the video at the bitrate $R$. A data path was defined as a logical route from the source to a node in the tree. When leeches block all the data paths in the overlay, no new peers can join the system. A P2P system is said to suffer from *clogging* when it cannot accept any more peers. In this appendix, we examine how often system clogging occurs in a simple scenario, where a system with a single tree accommodates two classes of heterogeneous peers. This simple configuration effectively captures how the available system capacity affects clogging with respect to the group size and server bandwidth. In particular, the analysis presented below shows that (1) clogging is more probable when the group size increases; and (2) peer join sequences have a significant impact on the system throughput.

We define the system capacity as the aggregate of peer uplink bandwidths. Similarly, the system throughput is defined as the aggregate of peer uplink bandwidth used to relay video in the overlay. Suppose that two classes of heterogeneous peers arrive in the system. $\frac{N}{2}$ Class A peers have an uplink bandwidth of $2R$, and $\frac{N}{2}$ Class B peers have an uplink bandwidth smaller than $R$. Class A peers can have up to two child peers each, whereas Class B peers cannot have any child peers because a minimum bandwidth of $R$ is required to support one child peer. When a peer from either class arrives at the system, the peer finds a parent peer with available bandwidth if
one exists. In Figure B.1, a peer join sequence is modeled as a random walk. The horizontal axis shows peers’ arrival order. The vertical axis represents the system’s total bandwidth available for new peers. The unit of the vertical axis is the number of new peers that can be accepted. The random walk starts at \((0, B_0)\) where \(B_0\) denotes the source peer’s bandwidth. An up-step occurs when a class A peer arrives; the class A peer consumes one unit of the system resource, while providing two units. Thus, a class A peer entails a net gain of one unit. A down-step occurs when a class B peer arrives; it consumes one unit. The random walk finishes at \((N, B_0)\) because the net contribution of \(N\) peers is 0. System clogging occurs when the random walk hits the horizontal axis.

Figure B.1: A random walk model for total available uplink bandwidth in the overlay. \(B_0\) is the number of peers that the source peer can directly serve. Left: Example of good path. Right: Example of bad path. The reflected segment starting from \(-B_0\) is depicted with a dotted line.

We apply the Ballot Theorem [150] with some modification to compute the likelihood of system clogging. In the original Ballot Problem, an election is considered in which candidate A receives \(a\) votes and candidate B receives \(b\) votes, where \(a \geq kb\) for some positive integer \(k\). The theorem states how to count the ways in which the ballots can be ordered so that A maintains more than \(k\) times as many votes as B throughout the counting of the ballots. For our problem, \(k\) is set to 1 and \(a = b = N/2\). Unlike in the original Ballot Problem, we set the initial point to \(B_0\). We compute the percentage of paths that stay above the horizontal axis among all of the possible paths; these are called good paths (See Figure B.1). Instead of counting good paths, however, we choose to count bad paths using the “reflection method.” In
APPENDIX B. ANALYSIS OF SYSTEM CLOGGING

the reflection method, the first down-step that meets the horizontal axis is identified. Then, the initial portion of the path, between the starting (leftmost) point and the identified point, is reflected across the horizontal axis. As illustrated in Figure B.1, every bad path has a reflection path that starts at \((0, -B_0)\) and ends at \((N, B_0)\). Moreover, every path from \((0, -B_0)\) to \((N, B_0)\) has a corresponding path from \((0, B_0)\) to \((N, B_0)\) that touches or crosses the horizontal axis at least once. Based on this one-to-one mapping relationship, we conclude that the total number of bad paths is the same as that of all paths from \((0, -B_0)\) to \((N, B_0)\). There are \(\left(\frac{N}{2} + B_0\right)\) up-steps and \(\left(\frac{N}{2} - B_0\right)\) down-steps in each bad path. Assuming \(N > 2B_0\), the number of all bad paths is obtained as \(\binom{N}{\frac{N}{2} + B_0}\). Then, the probability of system clogging is represented as the ratio of the number of bad paths to the number of total paths:

\[
\Pr(\text{system clogging}) = \frac{\text{Number of bad paths}}{\text{Number of total paths}} = \frac{\binom{N}{\frac{N}{2} + B_0}}{\binom{N}{\frac{N}{2}}} = \prod_{i=1}^{B_0} \frac{\frac{N}{2} - i + 1}{\frac{N}{2} + i}.
\]

(B.1)

Figure B.2 shows the probability of system clogging depending on the bandwidth of the source \((B_0)\) and the number of peers \((N)\). From the figure along with our combinatorial analysis of the example above, we draw the following conclusions: (1) the probability of clogging decreases when \(B_0\) increases; (2) clogging is more probable when the system size \(N\) increases even when \(B_0\) is high; and (3) peer join sequences have a significant impact on the actual system throughput. From our findings, we conjecture that clogging is inevitable in P2P systems serving peers that have more than two types of uplink bandwidths.
Figure B.2: Probability of system clogging under incremental overlay construction. Source uplink bandwidth is the number of child peers the source peer can directly serve.
Appendix C

Modeling Self-Sustainability Likelihood

In this appendix, we build a mathematical model to deepen our understanding about the influence of $\alpha$, $\beta$, and $\gamma$ on the self-sustainability likelihood discussed in Section 3.5.3. The definitions of the symbols can be found in Section 3.5.3.

C.1 Approximate Model

Our approach first reduces the number of random variables in (3.13) by the following approximation; let us expand $P(S)$ in (3.13) by conditioning on $A_C \geq N_C$:

$$
P(S) = P(A_C + A_B + B_B \leq J, A_B + B_B \geq N_B, A_C \geq N_C),
= P(A_C \geq N_C)P(N_B \leq A_B + B_B \leq J - A_C | A_C \geq N_C). 
$$

(C.1)

Let $O_C = A_C$ and $O_B = A_B + B_B$, where $O_X$ denote the number of outdegrees assigned to Type $X$. (C.1) is then rewritten as $P(O_C \geq N_C)P(N_B \leq O_B \leq J - O_C | O_C \geq N_C)$. When $P(O_C \geq N_C) \approx 1$ and $P(N_B \leq O_B \leq J - O_C | O_C \geq N_C)$ is not very sensitive to $O_C$, we assume that $O_B$ and $O_C$ are independent of each other.
$P(S)$ is then approximated as

$$
P(S) \approx P(O_C \geq N_C)P(N_B \leq O_B \leq J - \tilde{O}_B),
$$

(C.2)

where $\tilde{O}_B$ is a constant that replaces $O_C$ in (C.1). Although $A_B(= O_B - B_B)$ and $A_C(= O_C)$ have a mild negative correlation, this approximation holds well in the ranges of $A_B$ and $A_C$ we are interested in. We set $\tilde{O}_B = N_C$, the minimum value of $O_C$ in a subset of the sample space where $O_C \geq N_C$. Because this assignment allows for $O_B + O_C > J$, the second term of (C.2) may be overestimated when the region above the line $O_B = - O_C + J$ in the $O_C$-$O_B$ plane has a non-negligible joint distribution.

The distribution of $O_C$ is equivalent to the marginal distribution of $A_C$ in the joint distribution $P(a_A, a_B, a_C)$ in (3.9), which is a binomial distribution with parameters $D_A$ and $\alpha$. When $D_A$ is large and $D_A\alpha(1 - \alpha)$ is relatively large (e.g., over 10), then by the DeMoivre-Laplace limit theorem, we can approximate $P(O_C)$ to the normal approximation with mean $\mu_{O_C} = D_A\alpha$ and variance $\sigma_{O_C}^2 = D_A\alpha(1 - \alpha)$. The solutions in Table 3.5 suggest that the maximum $P(S)$ occurs when $N_C + m\sigma_{O_C} = \mu_{O_C}$, where $1.5 \leq m \leq 2.5$. Although optimal $\alpha$ values are determined by peer characteristics, this relation provides good guidance to the range of $\alpha$. $A_B$ follows a binomial distribution with parameters $D_A$ and $\beta$, and $B_B$ follows a binomial distribution with parameters $D_B$ and $\gamma$. Since $A_B$ and $B_B$ are independent of each other, their sum, $O_B$, also approximates to the normal distribution with mean $\mu_{O_B} = D_A\beta + D_B\gamma$ and variance $\sigma_{O_B}^2 = D_A\beta(1 - \beta) + D_B\gamma(1 - \gamma)$. Then, the two terms in (C.2) can be written as

$$
P(O_C \geq N_C) = P(O_C \geq N_C - 0.5)
= P \left( \frac{O_C - \mu_K}{\sigma_K} \geq \frac{N_C - 0.5 - \mu_K}{\sigma_K} \right)
\approx 1 - \Phi \left( \frac{N_C - 0.5 - \mu_K}{\sigma_K} \right),
$$

(C.3)
\[ P(N_B \leq O_B \leq J - N_C) = P(N_B - 0.5 \leq O_B \leq J - N_C + 0.5) \]
\[ = P \left( \frac{N_B - \mu_O - 0.5}{\sigma_O} \leq \frac{O_B - \mu_O}{\sigma_O} \leq \frac{J - N_C - \mu_O + 0.5}{\sigma_O} \right) \]
\[ \approx \Phi \left( \frac{J - N_C - \mu_O + 0.5}{\sigma_O} \right) - \Phi \left( \frac{N_B - \mu_O - 0.5}{\sigma_O} \right) \tag{C.4} \]

where \( \Phi(\cdot) \) is the standard normal CDF.

We can show that both (C.3) and (C.4) increase as the group size \( N \) increases. Since the normal distribution is symmetric around its mean, a high probability in (C.4) occurs when \( \mu_O \) is set to \( J - N_C + N/2 \). By substituting this into the definition of \( \mu_O \) and arranging it with respect to \( \gamma \), we get

\[ \gamma = -\frac{D_A}{D_B} \beta + \frac{J - N_C + N}{2D_B}. \tag{C.5} \]

Eq. (C.5) implies that the likelihood of self-sustainability is high along a line in the \((\beta, \gamma)\) spaces, as was shown in Fig. 3.19. The variance \( \sigma_O^2 \) can be written as a negative quadratic function of \( \gamma \) for a certain range in \((0,1)\), having its lowest values near \( \gamma = 0, 1 \). Thus, we see two local maxima along the line given in (C.5) in the \((\beta, \gamma)\) space as well as in Fig. 3.19. The model we developed in this section allows us to understand the system’s self-sustainability with respect to the peer characteristics and system parameters. When \( P(S) \) is close to 1, the model estimates the influence of the parameters with a high accuracy. The model predicts that higher self-sustainability can be achieved by a larger number of peers and higher average outdegrees. The model also reveals that high self-sustainability is achieved when Type A supports as few Type B peers as possible, which reduces the uncertainty in the number of outdegrees reserved for Type B.
C.2 Probabilistic Study

In Fig. C.1 to C.4, the models in (C.3) and (C.4) were compared with the empirical distributions of \( P(O_C \geq N_C) \) and \( P(N_B \leq A_B + B_B \leq J - A_C|A_C \geq N_C) \) in (C.1). In the comparison, the parameters of Case 1 in Table C.1 and three points (the two local maxima, \( \alpha = 0.33, \beta = 0.15, \gamma = 1 \) and \( \alpha = 0.33, \beta = 0.67, \gamma = 0.22 \), and their middle point \( \alpha = 0.33, \beta = 0.41, \gamma = 0.61 \)) were used. For the empirical distributions, we computed the distributions based on the computer-generated random numbers. Fig. C.1 shows the \( O_C \) probability distribution for the three points that shared the same \( \alpha \). Fig. C.2, C.3, and C.4 show the \( O_B \) distributions for the three points. Overall, the models predicted the empirical distributions with a high accuracy. The model for the \( O_B \) distribution in case of the middle point slightly deviated from the empirical distribution partly because the \( O_C, O_B \) joint distribution has non-negligible region above the line \( O_B = -O_C + J \), thus overestimating \( P(S) \).

Table C.1: Peer characteristics. \( N \) is the total number of peers. The peer type distribution: \( P(E_A) = a, P(E_B) = b, \) and \( P(E_C) = c \). \( d_X \): average out-degree of Type \( X \). \( O_R \): over-provisioning factor.

<table>
<thead>
<tr>
<th></th>
<th>( N )</th>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d_A )</th>
<th>( d_B )</th>
<th>( d_C )</th>
<th>( O_R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>300</td>
<td>0.423</td>
<td>0.424</td>
<td>0.153</td>
<td>1.35</td>
<td>0.87</td>
<td>1.08</td>
<td>1.1</td>
</tr>
</tbody>
</table>

\(^1\)The empirical \( P(S) \) was 0.764, whereas the model \( P(S) \) was 0.780. As a comparison, the difference between the empirical \( P(S) \) and model \( P(S) \) was less than 0.003 for the two local maxima.
Figure C.1: Distribution of $O_C$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_C} = 56.4$, $\sigma_{O_C} = 6.14$). No visible differences were found for various $\beta$ and $\gamma$ values.

Figure C.2: Distribution of $O_B$ for $\alpha = 0.33, \beta = 0.15, \gamma = 1$. $P(S) = 0.903$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_B} = 135.65$, $\sigma_{O_B} = 4.67$).

Figure C.3: Distribution of $O_B$ for $\alpha = 0.33, \beta = 0.41, \gamma = 0.61$. $P(S) = 0.765$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_B} = 135.62$, $\sigma_{O_B} = 8.27$).

Figure C.4: Distribution of $O_B$ for $\alpha = 0.33, \beta = 0.67, \gamma = 0.22$. $P(S) = 0.880$. The bars show the empirical probability mass function. The solid curve shows the model ($\mu_{O_B} = 138.77$, $\sigma_{O_B} = 7.52$).
Appendix D

A Measurement Study of Peer Properties

One of the difficulties in building a scalable and reliable peer-to-peer system is limited knowledge about peer characteristics. In this appendix, we present a rich set of peer characteristics based on our experiences with deploying a commercial variant of SPPM for the ESWC, a large-scale streaming event\(^1\). During this event, about 33,580 unique viewers from 93 countries watched the live video. We collected empirical data including user behavior and characteristics. Section D.1.1 provides an overview of the system used in the experiments. In Section D.1.2, we describe the experiments and methods for data collection. Section D.2 presents the analysis of the collected data.

D.1 Measurements

D.1.1 System Configuration

Fig. D.1 illustrates an overview of SPPM with enhancement to achieve commercial grade streaming service. It consists of a real-time video encoder, a session manager, a retransmission server, and a population of peers. Peers comprise super nodes and

\(^1\)ESWC, standing for Electronic Sports World Cup, is an international professional gaming championship [5].
Figure D.1: An example system configuration: video is captured and encoded in real time. A session manager receives the encoded signal and delivers it to peers using the SPPM protocol. Peers consist of a set of super nodes and a group of users. Super nodes are special servers that can provide additional relaying capacity to the system. From the protocol’s point of view, they act as normal peers with high uplink bandwidth, which are willing to relay video to other peers. The session manager constantly monitors the status of a session, including the aggregate of the uplink bandwidths. When the aggregate bandwidth is not sufficient to support an entire peer population, the super nodes provide their uplink bandwidths to the system.

During a live session, a video signal was encoded in real time at the venue of the event. The encoded video stream was transmitted to the session manager. The session manager acted as a video source in the multicast session as if it were the origin of the video. For video distribution, eight complementary trees were built as the data distribution overlay. To serve missing or late packets, a retransmission server was placed in each session. In the original design of SPPM, retransmissions were handled only locally among adjacent peers in the overlay. In the design used for this measurement, retransmission servers complement the local retransmission by providing reliable retransmissions of video packets that were close to their playout deadline.
D.1.2 Data Collection

We collected data from the ESWC 2008, held in San Jose, California, USA, from the 25th to the 27th of August [5]. The event was a world-wide PC game tournament hosted annually by ESWC. It was broadcast live using the system described in Section D.1.1. A variety of online PC game matches were broadcast for 5 to 12 hours every day. Each match lasted around 30 minutes to 1 hour. The video was encoded using H.264/AVC at 10 frames per second with an intraframe period of 20 frames. To avoid additional encoding latency, B frames were not employed. The spatial resolution was 640 by 480 pixels. No protection, such as FEC, was added to the bitstream. Instead, robustness to packet loss and/or peer churn was provided by retransmission. A mono channel of the audio was sampled at 44.1 kHz, 16 bits/sample and compressed using AAC. The total bitrate was approximately 600 kbps (the video accounted for 560 kbps and the audio for 40 kbps). Most of the video scenes were computer-generated, fast-moving 3D graphics. In general, the visual quality was acceptable when the packet loss ratio (PLR)\(^2\) was less than 2%.

Peers reported basic statistics, such as PLR and connection duration to the session manager at regular intervals. Retransmitted packets were considered to be received on time as long as they arrived before their deadline. We collected peer reports and the general session information generated by the session managers, including the size of the peer population, session length, and the average PLR.

D.2 Analysis

In this section, we analyze the data obtained from ESWC 2008. Individual statistics for each day are presented since averaging across sessions might not highlight peculiarities of each session.
D.2.1 Subscription Behavior

First, we present peers’ subscription behavior based on peer join and leave in a system. Temporal evolution of the number of concurrent users, shown in Fig. D.2, captures the group-wise dynamics due to peer join and leave. The peaks in the graphs were usually followed by a sudden drop due to the simultaneous departure of users who, most likely, had finished watching the matches they were interested in. On Day 2, there was a brief system outage around Hour 9. Around this time, the number of connected users dropped to nearly 0. After the system became available, about half of the disconnected users rejoined it in the next few minutes.

Fig. D.3 graphs the probability mass function of peer lifetime, the duration between the time a peer joins the system and the time it leaves the system. About half of the users left the system within five minutes after they joined. This indicates that peers are highly volatile, which is a primary root cause of unreliability in the system. In Fig. D.4, two curves based on the two well-known probability distributions, the exponential and the Zipf distribution, are plotted together with the data curve. The exponential distribution is popular in simulations and in the mathematical analysis of P2P system performance due to its simplicity and special properties, such as memorylessness. The Zipf distribution, known for its long tail\(^3\), is examined because peer

---

\(^2\)PLR is the ratio of the number of missing/late video packets to the total number of video packets.

\(^3\)Since the distribution based on the observations was represented as a PMF, we selected the Zipf
Figure D.3: The probability mass functions (PMF) of peer lifetime for each day.

Figure D.4: The PMFs of peer lifetime. Two distributions are fitted against data.

Lifetime is known to exhibit a long tail distribution [49, 111]. The data were fitted with the two distributions. In Fig. D.3, the exponential distribution had a mean of 8.0 minutes, which was much smaller than the actual average of 27.1 minutes. The Zipf distribution had a parameter of $s = 1.59$, which is expressed as

$$f(X = x) = \frac{1/x^{1.59}}{\sum_{n=1}^{N} (1/n^{1.59})},$$

(D.1)

where $X$ takes on the medium value of each 5-minute-long interval. As pointed out in prior work, the Zipf distribution fits the observations better than the exponential distribution. Another implication of the Zipf distribution is the strong correlation between a peer’s instantaneous connection duration and its lifetime. To evaluate this correlation, we considered the conditional probability that a peer may leave the system during the next 30 minutes\(^4\) given that the peer has been present in the system for $t$ minutes. Fig. D.5 illustrates the conditional probability in (D.2) for the data and two models.

$$\Pr(T < t + 30 | T \geq t),$$

(D.2)

where $T$ denotes a peer’s connection duration and both $T$ and $t$ are in minutes. The distribution in lieu of the Weibull distribution.

\(^4\)A large period (of 30 minutes) was chosen to smooth out the curve obtained from the empirical data.
memoryless property of the exponential distribution implies that a peer’s departure probability during the next 30 minutes is 97%, regardless of its connection duration. However, the data show that the actual departure probability was much lower than 97% and invalidated the memoryless property with a monotonous decrease over time. The Zipf distribution model shows a similar trend as the data. The lesson we learned from the correlation between connection time and lifetime can be exploited in the construction of overlays: since peers who have recently joined the system have a high departure probability, young peers should not be promoted toward the tree root too early. Also, when peers look for other peers to get video, they may prefer peers who have been in the system longer in order to reduce parent disconnects.

![Figure D.5: Conditional distribution of the departure of an active peer within the next 30 minutes.](image)

**D.2.2 Connectivity Restriction**

Next, we study the accessibility of peers, which affects peer-wise connection in a practical P2P system. When an SPPM client joined the system, it discovered the existence of a NAT (or a firewall), and the type of NAT, by running a variant of the STUN algorithm [152]. We classified peers according to four different NAT-type
categories—full cone, restricted IP, restricted port, and symmetric NAT—and according to two non-NAT type categories—static IP and static IP firewall. Fig. D.2.2 depicts the distribution of the NAT type of peers. We observed no distinct difference across the days, indicating that there is a weak stationarity in the distribution of peer NAT types. The majority of peers turned out to be restricted-port NAT type, accounting for half of the peer population. In Fig. D.6, the peer contributions of uplink bandwidth are depicted according to their NAT type. The figure shows that the ratio of the aggregate bandwidths contributed by peers without NAT to the total aggregate bandwidths is higher than the ratio of the number of corresponding peers to the population. A plausible explanation is that a large number of peers without NAT are part of a high-bandwidth network, such as a university campus or research institution.

NATs usually block connection attempts from outside the network. Peers with static IP addresses, but behind a firewall, are not visible to the outside unless they first initiate communication with other peers. Peers with full-cone NAT are accessible from outside once their public address assigned at the NAT device is known. Peers with static IP and peers with full-cone NAT are most easily accessible in a P2P system. Peers with restricted-IP NAT accept traffic only from the network addresses (regardless of transport layer port number) with which they have initiated communication. Thus, both peers with restricted-IP NAT and peers with static IP, but behind firewalls, become accessible to external peers only if they first initiate communication with them. Peers with restricted-port NAT accept only external traffic with a network address and a transport layer port number with which they have initiated communication. Peers with symmetric NAT are assigned a different pair of address and port for each connection they establish with external computers. Peers in this NAT category have the most limited accessibility. These limitations in peer-wise connection due to NATs and firewalls can pose a serious difficulty in overlay construction [75].
However, with external assistance, such as UDP hole punching[73], peer-wise connection can be substantially improved, thus increasing the aggregate uplink capacity of a P2P system. Table D.1 lists peer-wise connection between different NAT types of peers with the help of the UDP hole punching technique.

<table>
<thead>
<tr>
<th></th>
<th>Static IP</th>
<th>Full cone</th>
<th>Rest. IP</th>
<th>Static f/w</th>
<th>Rest. Port</th>
<th>Symm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static IP</td>
<td>○</td>
<td>○</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>Full cone</td>
<td>○</td>
<td>○</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>Restricted IP</td>
<td>○</td>
<td>○</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>Static f/w</td>
<td>○</td>
<td>○</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>Rest. Port</td>
<td>○</td>
<td>○</td>
<td>△</td>
<td>△</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Symmetric</td>
<td>○</td>
<td>○</td>
<td>△</td>
<td>△</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

D.2.3 Geographical Distribution

In Fig. D.7, we present the geographical distribution of users. All connections were counted separately based on their IP address and port combination; if the same user connected to the system several times on a day, then these connections were counted separately. Users from 93 countries participated in the event online. The distribution of users’ geographic location reflects the Pareto principle (the 80/20 rule) in that 84.7% of the users come from 20% of the countries (18 countries). The fact that the majority of users are located closely to each other shows that P2P system designers should take user proximity into consideration in designing a P2P system [202].

![Figure D.7: User’s geographic distribution (all days). Users came from 93 countries.](image)

---

5 “The Country WhoIS DB” (July 10, 2008 release) available from TamoSoft [12] was used to map an IP address to its geographical location.
D.2.4 Uplink Contribution

The distribution of peer uplink bandwidth is shown in Fig. D.8. More than half of the users contributed less than 600 kbps of uplink bandwidth, where 600 kbps was the total streaming bitrate. Since each multicast tree delivered a fraction of the stream, peers with uplink bandwidth lower than the total bitrate were still able to contribute to the aggregate system capacity. The average uplink bandwidth was 1647 kbps, much higher than the video bitrate contributed by high-profile peers. As other researchers have pointed out [111], an effective treatment of high-profile peers is critical in improving the relaying capacity of the system.

![Peer uplink bandwidth distribution](image)

Figure D.8: Peer uplink bandwidth distribution.

To see the impact of external assistance on peer-wise connection, we consider the following three scenarios:

- **Theoretical upper limit**: average uplink bandwidth contributed by a peer in case of no restriction in peer-wise connection. It provides the upper bound of peer contribution.

- **With assistance**: average uplink bandwidth contributed by a peer if external assistance is employed. Besides static IP and full-cone NAT peers, peers behind the other types of NATs or firewalls can contribute their bandwidth by
employing UDP hole punching [73]. Note that the hole punching technique fails to provide connectivity between some pairs of NAT types (see Table D.1).

- Without assistance: average uplink bandwidth contributed by a peer when no external assistance is provided. Only static IP and full-cone NAT peers can contribute their bandwidth in this case.

These peer contributions are expressed in (D.3) and the notations are listed in Table D.2.

\[
\begin{align*}
N_{\text{all}} &= \sum_{\forall \text{type}} N_{\text{type}} \\
BW_{\text{all}} &= \sum_{\forall \text{type}} BW_{\text{type}} \\
C_{\text{wo}} &= \frac{BW_{\text{ST}} + BW_{\text{FC}}}{N_{\text{all}}} \\
C_{w} &= \left\{ BW_{\text{RI}} + BW_{\text{SF}} + BW_{\text{RP}} \cdot \left(1 - \frac{N_{\text{SM}}}{N_{\text{all}}}\right) + \\
&\quad BW_{\text{SM}} \cdot \left(1 - \frac{N_{\text{SM}} + N_{\text{RP}}}{N_{\text{all}}}\right)\right\} \cdot \frac{1}{N_{\text{all}}} + C_{\text{wo}} \\
C_{\text{theo}} &= \frac{BW_{\text{all}}}{N_{\text{all}}}. 
\end{align*}
\]

Fig. D.9 illustrates the average uplink bandwidth over time according to the different contribution scenarios. Overall, the assisted contribution was nearly 85% of the full contribution. This observation clearly emphasizes the benefit of external assistance in improving peer-wise connection. On Day 1 and 2, the fluctuation of the average uplink bandwidth was significant during the event. The high fluctuation and the uncertainty in peer contribution often results in a P2P system being forced to serve users less-than-ideal video quality. If the system is required to serve a consistently high quality of video, additional relaying capacity must be provided by dedicated relay servers, such as super nodes.
Table D.2: Symbols used in (D.3)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>Number of peers</td>
</tr>
<tr>
<td>$BW$</td>
<td>Total uplink bandwidth</td>
</tr>
<tr>
<td>$C_{wo}$</td>
<td>Contribution without assistance</td>
</tr>
<tr>
<td>$C_w$</td>
<td>Contribution with assistance</td>
</tr>
<tr>
<td>$C_{theo}$</td>
<td>Theoretical maximum contribution</td>
</tr>
<tr>
<td>$ST$</td>
<td>Static IP</td>
</tr>
<tr>
<td>$SF$</td>
<td>Static IP behind firewall</td>
</tr>
<tr>
<td>$FC$</td>
<td>Full-cone NAT</td>
</tr>
<tr>
<td>$RI$</td>
<td>Restricted-IP NAT</td>
</tr>
<tr>
<td>$RP$</td>
<td>Restricted-port NAT</td>
</tr>
<tr>
<td>$SM$</td>
<td>Symmetric-NAT</td>
</tr>
</tbody>
</table>

Table D.3: Average uplink bandwidth contributed by a peer based on contribution scenario.

<table>
<thead>
<tr>
<th></th>
<th>theoretical</th>
<th>with assistance</th>
<th>without assistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>1647 kbps</td>
<td>1385 kbps</td>
<td>672 kbps</td>
</tr>
<tr>
<td>Day 2</td>
<td>1530 kbps</td>
<td>1283 kbps</td>
<td>627 kbps</td>
</tr>
<tr>
<td>Day 3</td>
<td>1141 kbps</td>
<td>959 kbps</td>
<td>414 kbps</td>
</tr>
</tbody>
</table>

D.3 Summary

In this appendix, we have presented a number of peer characteristics that are crucial in peer-to-peer (P2P) system design. First, we discussed peer’s subscription behavior, which consists of peer arrival and departure pattern. In particular, we showed that peers are volatile – about 50% of peers left the system less than five minutes after they joined it. Next, we studied the connection restriction of peers due to the existence of NAT devices and firewalls. Taking into consideration the existence of NATs and firewalls, we showed that peers can contribute sufficient uplink bandwidth to the system if peer-wise connection is improved with external assistance. The analysis also revealed that the aggregate of peers’ uplink bandwidths fluctuated widely during
the session. It suggests that a P2P streaming system should be designed to be flexible, so that the streaming quality adapts to the user population at a given moment.

Finally, the geographical distribution of peers reveals that about 85% of peers are geographically located in 20% of the countries, and the 70% of relaying capacity was supplied by 20% of peers. Therefore, one should consider the 80/20 rule when designing P2P systems.
Figure D.9: Temporal evolution of the average uplink bandwidth. Left: Day 1, Middle: Day 2, Right: Day 3.
Bibliography


