Adaptable Peer-to-Peer Internet Live Media Streaming

Bartosz Biskupski

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

March 2009

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work. I agree that Trinity College Library may lend or copy this thesis upon request.

Bartosz Biskupski

Dated: 13th March 2009

Acknowledgements

I wish to express my gratitude to my supervisor, Dr. René Meier, for his guidance and valuable comments to this thesis. Many thanks also to Dr. Jim Dowling for advising me during the early stages of my PhD studies.

To amazing friends and colleagues in the Distributed Systems Group and especially to Raymond Cunningham, Serena Fritsch, Marcin Karpiński, Jan Sacha, As'ad Salkham, Tim Walsh, Stefan Weber, and friends related to this group: Sanad Gharaibeh, Gosia Jaksik, Jean Joswig, for standing by me the whole time, for support when needed, and for enjoying the good times with me. Thank you all for the interesting and fun conversations, great trips, nights out, lunches, and tea/coffee breaks.

To Paweł Garbacki, Jan Sacha and Marc Schiely for research collaboration and to Raymond Cunningham for research collaboration and proof-reading parts of this thesis.

To my PhD examiners, Alberto Montresor and Declan O'Sullivan, for useful comments on this thesis.

To all my other friends: those in Ireland for the great time over the last four years, those in Poland for making my each visit home much too short, and those around the world for staying in touch.

To my Mom, my Dad, my sister Kamila, my mom-in-law Renia, and the rest of my family and family-in-law for being the best family ever. And most of all, to my love and wife, Ilona, for enjoying life together with me and her great patience during this PhD journey.

Abstract

Media streaming is an approach to delivering media, which may consist of video and audio, from a provider to viewers. Media streaming enables simultaneous delivery and playback of media and thus provides an alternative to media download, where the entire media content has to be delivered before the playback can begin. Media streaming can be on-demand for content archived at the provider or live for content produced at the time of delivery.

Live media streaming does not scale with current client-server-based approaches due to large bandwidth and server requirements of the content provider. IP multicast has been proposed as a network-level approach for multiple users to concurrently receive content transmitted from the server. However, limited deployment of IP multicast prevents pervasive use for live streaming. As a result, there is a growing interest in application-level peer-to-peer approaches that do not require specific support from the network. These approaches use bandwidth of viewers, called peers, to reduce bandwidth and server consumption of the content provider.

In this thesis, we present the MeshTV peer-to-peer system for live media streaming over the Internet. MeshTV addresses limitations of existing peer-to-peer live streaming systems that adapt poorly to the dynamic and heterogeneous nature of the Internet environment. MeshTV uses a mesh-based approach to peer-to-peer live media streaming. In mesh-based approaches, every peer is connected to multiple other peers, forming a mesh overlay. A distinguishing characteristic of a mesh overlay is that it supports

connections between any two peers. This simplifies the adaptation of the overlay to arrival and departure of peers and to fluctuations in peer bandwidth.

MeshTV optimises the quality of media playback at participating peers. It uses a novel decentralised algorithm for adapting the mesh overlay. The overlay adapts so that the entire heterogeneous outgoing bandwidth of peers is utilised for media streaming. The overlay also adapts so that all peers download media content at approximately the same rate, unless their incoming bandwidth reduces their download rate. To adapt the quality of media playback at a peer to the download rate of the peer, MeshTV encodes the original media stream into multiple media descriptions. Every subset of these descriptions can be decoded for playback and the quality of playback corresponds to the number of descriptions being used for decoding. Peers use an algorithm to adapt the number of downloaded descriptions to their download rate.

MeshTV furthermore enables a short delay between selecting a media for playback and the start of the actual playback. This playback startup delay is required to buffer a sufficient amount of content to provide continuous playback in light of departure of peers, variations in download rates and non-sequential delivery of content. A peer in MeshTV initially delivers playback of a basic quality that allows for a short startup delay. The quality of media playback is then gradually improved over time.

This thesis presents a comprehensive simulation analysis of MeshTV. Evaluation results show that MeshTV adapts the overlay so that the upload rate of peers is maximised and download rates are nearly uniform among peers. The time of this adaptation is short and independent of the number of peers in the overlay. The quality of playback at peers is shown to adapt to their respective download rate. The results show that MeshTV is resilient to highly transient population of peers. Moreover, the results show that viewers may watch playback of a basic quality incurring only 3 seconds of startup delay.

Publications Related to this Ph.D.

- B. Biskupski, M. Schiely, P. Felber, and R. Meier. Tree-based Analysis of Mesh Overlays for Peer-to-Peer Streaming. In *8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'08)*, LNCS, Oslo, Norway, June 2008.

- R. Cunningham, B. Biskupski, and R. Meier. Managing Peer-to-Peer Live Streaming Applications. In *8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'08)*, LNCS, Oslo, Norway, June 2008.

- B. Biskupski, R. Cunningham, and R. Meier. Improving Throughput and Node Proximity of P2P Live Video Streaming through Overlay Adaptation. In *International Symposium on Multimedia (ISM'07)*, IEEE, Taichung, Taiwan, December 2007.

- B. Biskupski, J. Dowling, and J. Sacha. Properties and Mechanisms of Self-Organising MANET and P2P Systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS'07)*, ACM Press, March 2007.

- J. Sacha, B. Biskupski, D. Dahlem, R. Cunningham, J. Dowling, and R. Meier. A Service-Oriented Peer-to-Peer Architecture for a Digital Ecosystem. In *1st IEEE International Conference on Digital Ecosystems and Technologies (DEST'07)*, IEEE, Cairns, Australia, February 2007.

- D. Dahlem, L. Nickel, J. Sacha, B. Biskupski, and R. Meier. Towards Improving the Availability of Service Compositions. In *1st IEEE International Conference on Digital Ecosystems and Technologies (DEST'07)*, IEEE, Cairns, Australia, February 2007.

- B. Biskupski, R. Cunningham, J. Dowling, R. Meier. High-Bandwidth Mesh-based Overlay Multicast in Heterogeneous Environments. In *2nd International*

Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA 2006), ACM Press, Pisa, Italy, October 2006.

- D. Dahlem, D. McKitterick, L. Nickel, J. Dowling, B. Biskupski, R. Meier. Binding- and Port-Agnostic Service Composition using a P2P SOA. In *ICSOC Workshop on Dynamic Web Processes (DWP)*, Amsterdam, The Netherlands, December 2005.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A key challenge for the Internet infrastructure has been to deliver increasingly large and complex content to a growing Internet user population. Content offered on the Internet has evolved from small size text-only web pages to large size multimedia, such as audio and video. Approaches to viewing Internet media has also evolved from media download to media streaming. While media download typically requires that the entire media content is downloaded before the playback can begin, streaming enables simultaneous download and playback of media content by viewers.

Media streaming can be *on-demand* or *live*. On-demand streaming is used for delivery of archived media files to viewers for instant playback. In on-demand streaming viewers may request a media file whenever they want. Thus, different viewers may be downloading and watching different parts of the same media file at the same moment in time. Live streaming is used for delivery of "live" broadcasts, where media content is recorded and immediately sent out to all viewers. Thus, all viewers watch the same media content at the same time. For example, thousands of viewers may be watching a live stream of a sporting event.

To support on-demand or live streaming, a viewer needs to download individual data packets, which constitute media content, before their playback time. In particular,

data packets need to be downloaded by the viewer at a rate sufficient for playback and without interruptions. Congestion is a typical reason for data packets being undelivered before their playback time. Congestion occurs when the volume of content that needs to be sent exceeds the capacity of the server or the network infrastructure, such as network routers or links. When congestion occurs, some data packets may be lost or delayed. Lost packets may be retransmitted, but may not be received by a viewer before their playback time. To increase the likelihood of receiving packets before their playback time, playback at viewers is delayed with respect to the transmission of media content by the content provider. If, in spite of this delay, a packet is not received before its playback time, a viewer can either further delay the playback until the missing packet arrives or skip the playback of the missing packet.

To prevent congestion, streaming requires that servers and network infrastructure can support the amount of bandwidth sufficient to deliver media content to all viewers. Bandwidth describes how much data, which include the actual content and control data, can be transmitted in a defined time over a connection and is usually measured in bits per second (bps). *Uplink* and *downlink* bandwidth are two terms that describe the maximum rate at which an Internet host can, respectively, transmit and receive data. They rely on a common assumption that bandwidth is typically limited by Internet Service Providers at access links to the Internet rather than in the core of the Internet [85].

Current approaches for media streaming generally require a one-to-one unicast transmission from a server, or a set of servers, to each viewer. Thus, the amount of uplink bandwidth required for streaming corresponds to the number of viewers and the desired quality of media content. For a large number of viewers and high-quality media content, streaming poses great challenges. For example, in March 2006, the CBS live online broadcast of the NCAA basketball tournament attracted at its peak 268 000 simultaneous viewers [72]. Even with today's low-bandwidth low-quality video

streaming at 400 Kbps, the CBS broadcast needed over 100 Gbps of bandwidth. In comparison, the largest Content Delivery Network (CDN) provider, Akamai [3], at the time, reported a peak total capacity of 200 Gbps using tens of thousands of servers [72]. Forthcoming television broadcasts over the Internet are envisioned to attract even larger audience and require higher quality of media content. Current state-of-the-art video compression methods, such as AVC/H.264, require around 1.5 Mbps for the standard TV quality and around 6 Mbps for the High Definition TV (HDTV) quality for each viewer [86]. To reach 1 million of viewers with the standard TV quality broadcast, the aggregate 1.5 Tbps bandwidth capacity is required. This presents great challenges with currently available streaming technologies and motivates research towards more efficient content delivery over the Internet.

1.1 Internet Media Streaming

To offer media streaming over the Internet, a content provider may deploy a large number of dedicated servers that can support a large amount of bandwidth. However, building and managing such infrastructure is complex and involves significant costs. There is also a risk of under-provisioning or over-provisioning due to the difficulty of predicting the actual demand. When the number of viewers exceeds the capacity of the infrastructure, congestion occurs that results in viewers having to watch media at poor quality or even being prevented from watching. When the number of viewers is below the capacity of the infrastructure, network bandwidth and servers are unused. For these reasons, streaming media providers often use specialised commercial CDNs. CDNs maintain a large number of highly connected servers in multiple locations across the Internet and sell content hosting and distribution services to content providers. This relieves content providers from the burden of managing dedicated servers and provides a more fine-grained control over the capacity. Additionally, CDNs serve content from

servers nearest to the viewers. The content is sent over shorter network paths, thereby reducing the buffering delay, packet loss and the total Internet resource usage. Despite the advantages of CDNs, this approach to media streaming is expensive for content providers as CDN pricing is typically based on the amount of bandwidth used for streaming. Furthermore, the total capacity of existing CDNs may not be sufficient to stream high quality media content to a large number of viewers [72].

An approach to reducing bandwidth consumption is IP multicast [25], where the server needs to send out only one copy of a stream, while intermediate Internet routers supporting IP multicast take care of replicating this stream to all viewers. However, IP multicast has scalability and security issues that prevented its wide deployment on Internet routers [26]. Limited deployment of IP multicast prevents its use for live media streaming on a global scale.

1.2 P2P Content Delivery

Recent research in content delivery focuses on *peer-to-peer* (P2P) approaches. P2P systems are distributed systems consisting of user hosts, called *peers* or *nodes*, that are organised into a virtual network topology, called a *peer-to-peer overlay*, with the purpose of sharing their resources, such as content, storage space, computing power or network bandwidth [5]. In an overlay, each peer is connected to a small subset of the peers available in the overlay, called its *neighbours*.

Each peer can provide both client and server functionality, by both consuming and providing resources. This is in contrast to traditional client-server architectures, where resources are provided by servers and consumed by clients. While the performance and scalability of a client-server architecture is limited by the resources available at the servers, P2P systems can scale with demand. Each new peer consumes resources of other peers, but it also contributes its own resources. Furthermore, in contrast to

network-level approaches, such as IP multicast, P2P systems operate on an application-level and do not require any specific support from the underlying physical network. This enables fast deployment of P2P systems on the Internet as no changes to the existing Internet infrastructure are necessary.

P2P approaches have been used for distributing the storage space [24, 21] and processing power [108, 34] among participating peers. P2P approaches have attracted major attention in large-scale content distribution as network bandwidth is often the scarcest resource on the Internet. P2P approaches enable to significantly reduce the consumption of bandwidth at content providers by utilising bandwidth of participating peers. They are used in file download [22, 68], live media streaming [135, 17, 93, 89], on-demand media streaming [60, 38, 125], and voice-over-IP [9] applications. These application domains have different objectives and requirements, and thus require different P2P approaches. For example, the objective of file download is to download a complete file. The download rate may fluctuate and individual data packets may be received by a peer in any order. This is in contrast to live and on-demand media streaming that impose stringent constraints on the download rate and on the reception time of each packet. A peer needs to receive data packets before their playback time and at a rate that enables continuous playback. Live streaming has also different characteristics than on-demand streaming. In live streaming, a large number of viewers need to receive the same content at about the same time. This is in contrast to on-demand streaming, where playback of a media file is spread out over a long period of time, normally resulting in a smaller number of simultaneous viewers that need to receive different parts of the media file. Similar to media streaming, voice-over-IP applications have stringent constraints on the reception time of each data packet. Unlike media streaming, voice-over-IP does not need to support a large number of simultaneous participants or cope with the high-bandwidth requirements of media streaming.

1.3 P2P Live Media Streaming

In P2P live media streaming systems, the provider of media content, which we call a *transmitter*, sends content to only a small number of peers. Neighbouring peers in the P2P overlay forward content to each other, until all peers receive the content. This limits the consumption of bandwidth at the transmitter as bandwidth available from participating peers is utilised.

The two main P2P approaches to live streaming are *tree-based* and *mesh-based*. The tree-based approach organises peers into a single or multiple trees, where peers receive content from their parents and forward it to their children. However, tree-based approaches suffer from poor resilience to arrival and departure of peers, an occurrence called *peer churn*, and to fluctuations in peer bandwidth. In particular, departure of a peer that is an interior node in a tree results in the whole sub-tree rooted at this node ceasing to receive content until the tree is repaired. Furthermore, when the uplink bandwidth of an interior node decreases, the node may not be able to forward content to all its children and adaptation of the overlay may be necessary. Adaptation of the tree-structured overlay is complex as connections between peers must respect various constraints [76].

In mesh-based approaches, every peer is connected to multiple other peers, forming a mesh overlay. The media stream is split by the transmitter into small data chunks that are forwarded between neighbouring peers. This results in propagation of data chunks, throughout the mesh overlay, to all peers. A distinguishing characteristic of a mesh overlay is that it supports connections between any two peers. Thus, an arriving peer can connect to any other peers in the overlay. Similarly, a peer can replace a departing neighbour with any other peer. For these reasons, mesh overlays are highly resilient to peer churn. Furthermore, in mesh overlays, a peer does not depend on any particular neighbour to download data chunks. Thus, when the uplink bandwidth of a neighbour decreases, the peer may download data chunks from its remaining neighbours. For this

reason, mesh overlays are also highly resilient to fluctuations in the available bandwidth of peers.

1.4 Aims and Objectives

In this thesis, we present the MeshTV P2P system for streaming live media over the Internet. Aims and objectives of the MeshTV system are based on our intuition of what is important from the perspective of a viewer. These aims and objectives are:

Scalability. The performance of a scalable P2P system should not degrade when the number of peers increases. For this reason, coordination of a system should not be mediated by centralised components. Centralised components have finite resources that may become system bottlenecks and may limit system scalability. In P2P live streaming systems, centralised coordination may limit the maximum number of simultaneous viewers. Therefore, coordination should be decentralised, based on peers interacting directly with each other. In particular, maintenance and adaptation of a P2P overlay should be coordinated by individual peers, rather than by a centralised coordinator.

Resilience. P2P systems need to deal with highly dynamic P2P environments. P2P systems are subject to peer churn and fluctuations in the available peer bandwidth. If dissemination of media content depends on any specific structure of the P2P overlay, this structure may need to be constantly adapted to accommodate changes in peer population and peer bandwidth. Failure to adapt the overlay in a timely manner may result in packet loss and, ultimately, in interrupted playback at a large number of peers.

Optimal quality of media playback. Viewers expect to watch media content at a high quality. The quality of playback corresponds to the rate at which media content is encoded and delivered to a viewer. In P2P live streaming systems, this content

delivery relies on peers uploading content to each other. Thus, to maximise the quality of playback at all peers, the media content should be delivered to each peer at the same rate that maximises the upload rate of peers.

Furthermore, P2P live streaming systems should accommodate peers with various downlink bandwidth. Providing media content at the same rate to all peers may not be appropriate. A single media rate may exceed the downlink bandwidth of some peers and thereby may not allow for continuous playback at these peers.

Short playback startup delay. In contrast to traditional client-server architectures for live streaming, existing P2P live streaming systems suffer from a significant delay between the time when a viewer selects a media for playback and the start of the actual playback [72]. In client-server architectures, a viewer receives media content from a designated server, whereas in P2P live streaming systems, a viewer receives media content from peers that are ordinary personal computers. These computers may leave the P2P system at any time and their upload rate may fluctuate as multiple user applications may compete for the bandwidth. Additionally, many P2P live streaming systems are designed to deliver data packets to peers in a non-sequential order, whereas streaming requires sequential ordering. Thus, P2P live streaming systems tend to buffer much content to ensure continuous playback. This buffering causes startup delays and fails to provide channel-surfing experience of traditional television.

1.5 Contributions

MeshTV uses a mesh-based approach to P2P live media streaming for its high resilience to peer churn and to fluctuations in network bandwidth. The main contributions of this thesis include:

- We demonstrate that existing mesh-based approaches use only a portion of the

available peer uplink bandwidth for media streaming. This reduces the download rate of peers and thereby reduces the quality of their media playback. Moreover, we show that download rates are non-uniform among peers. When download rates are non-uniform among peers, it is inappropriate to provide media content at a single quality to all peers. Peers with the download rate below the media rate may not be able to deliver continuous playback. In turn, peers with the download rate above the media rate may deliver playback of suboptimal quality.

- MeshTV proposes a novel decentralised algorithm for adapting the mesh overlay. The overlay adapts so that the entire heterogeneous uplink bandwidth of peers is utilised for media streaming. The overlay also adapts so that all peers download media content at approximately the same rate, unless their downlink bandwidth reduces their download rate.

- To accommodate peers with the download rate reduced by their individual downlink bandwidth, MeshTV proposes algorithms for adapting the quality of playback at peers to their download rate. These algorithms also accommodate variations in the download rate of peers over time. These variations may be caused by peer churn and changing bandwidth of peers. To allow for the playback quality adaptation, the transmitter uses multiple description coding (MDC) technique [45] to encode media content. MDC produces multiple substreams, called descriptions, where any subset of these descriptions can be decoded. The quality of decoded media depends on the number of descriptions used for decoding. Peers in MeshTV continuously adapt the number of downloaded descriptions to their download rate.

- The MeshTV algorithms for adapting the quality of playback also reduce the playback startup delay. When a peer joins a P2P transmission, it initially downloads a single media description that corresponds to a basic quality of playback

and allows for a short startup delay. The basic playback quality is often sufficient for a viewer to decide whether to continue watching the transmission or to switch to a different one. The quality of media playback then gradually improves over time as the number of downloaded descriptions is increased.

A further contribution of this thesis is a packet-level network simulator that has been implemented for the analysis of MeshTV. This simulator strikes a balance between accuracy and performance of network modelling. For the accuracy of network modelling, it simulates transmission of every data packet over the network and imposes bandwidth and latency constraints of network links. For performance, it shares the available bandwidth of network links among connections competing for bandwidth without the overhead of simulating transmission of control packets for congestion avoidance and control. This allows to simulate P2P live streaming at high media rates in overlays consisting of 5000 heterogeneous and dynamic peers. In contrast, we were unable to simulate more than 500 peers using the well-known ns-2 simulator [84] running on the same hardware.

1.6 Outline

This thesis is organised as follows. In Chapter 2, we introduce approaches to large-scale and large-volume content delivery. We present background on P2P approaches to file download, live streaming and on-demand streaming. Chapter 3 reviews the current state-of-the-art related to adaptable P2P live media streaming and discusses its shortcomings. In Chapter 4, we describe the MeshTV system and its algorithms. In Chapter 5, we describe the network simulator and present a comprehensive simulation analysis of MeshTV. Finally, in Chapter 6, we conclude this thesis and discuss work that remains for future work.

Chapter 2

P2P Content Delivery

Delivery of large-volume content to a large number of users presents great challenges, mainly due to large bandwidth requirements. P2P systems are particularly attractive in this context as they can significantly reduce bandwidth requirements of a content provider by utilising bandwidth of participating users.

Section 2.1 introduces the main concepts of P2P systems. Subsequent sections introduce P2P approaches to large-scale and large-volume content delivery on the Internet. In particular, Sections 2.2, 2.3, and 2.4 present P2P approaches to file-sharing, live streaming, and on-demand streaming. Section 2.5 presents network-level multicast and discusses reasons that prevent its wide-scale adoption on the Internet. Finally, Section 2.6 draws similarities between the presented P2P approaches and concludes this chapter.

2.1 P2P Systems

P2P systems are distributed systems for sharing of computer resources, such as content, storage space, computing power or network bandwidth, by direct exchange, rather than requiring the intermediation or support of a centralised server or authority [5]. P2P

systems consist of *peers*, also called *nodes*, that are computers of individual users and that are logically interconnected with each other to form a *peer-to-peer overlay* on top of the physical network. In the P2P overlay, each peer is connected to a small subset of available peers, called its *neighbours*. Peers communicate with their neighbours using the underlying physical network.

In contrast to traditional client-server architectures, each peer in a P2P system can provide both client and server functionality by both providing and consuming resources of other peers. This enables the self-scaling property of P2P systems, where the available resources grow with the number of participating peers.

Peers are autonomous and may fail, join or leave the system at any time. A P2P system needs to accommodate such failures and transient population of peers, called *peer churn*, in order to maintain an acceptable connectivity and performance of the system.

A P2P overlay can be classified as *unstructured* or *structured*, based on the choice of neighbours of peers. In the following subsections, we discuss these two classes of P2P overlays.

2.1.1 Unstructured Overlays

An unstructured overlay does not rely on any specific connections between peers. Typically, any two peers can be connected with each other in an unstructured overlay. This simplifies adaptation of unstructured overlays to failure, arrival and departure of peers compared to structured overlays that restrict connections to specific pairs of peers. As a consequence, unstructured overlays demonstrate higher resilience to peer churn and lower overhead of overlay maintenance compared to structured overlays. However, the lack of a defined structure may result in a poor efficiency of some operations, such as searching for data in the overlay. Searching for data in an unstructured overlay typically requires that a query is flooded throughout the overlay, as in Gnutella [100]. This

incurs high communication overhead and does not guarantee that the query reaches a peer that has the requested data.

2.1.2 Structured Overlays

Structured P2P overlays require that connections between peers follow some predefined global pattern. The structure of the overlay is designed to provide high performance of some P2P operations, such as searching for data in the overlay or distributing content throughout the overlay. The most common type of a structured P2P overlay is the Distributed Hash Table (DHT). Similar to traditional hash table data structures, the DHT stores data composed of a key and a value and any participating peer can efficiently retrieve the value associated with a given key. Each peer is responsible for a part of an address space and stores all data with a key in this address space. Peer connections are designed so that each peer, for a given key, can efficiently locate the peer responsible for the part of the address space that contains this key. Some well-known DHT systems include Chord [116], Pastry [104], Tapestry [137], CAN [98] and Kademlia [79]. Another example of structured P2P overlays are tree-based overlays, which we discuss in Sections 2.3.1 and 2.3.2.

When peers fail, join, or leave the system, structured overlays typically require that the overlay is adapted respectively. For instance, in the DHT, peer churn may require that peers discover new neighbours to resume efficient lookup in the overlay. Due to the requirement of this, often complex, adaptation, structured overlays typically offer lower resilience to peer churn and higher overhead of overlay maintenance compared to unstructured overlays.

2.1.3 P2P Application Domains

P2P systems have been employed for a wide variety of application domains. Here, we show examples of how P2P systems are used to distribute the consumption of particular

resources in applications that require large amounts of these resources. P2P systems have been used for distributed computations by exploiting the computing power of large numbers of peers. A centralised server typically breaks down a computation intensive task into smaller work units and distributes them among participating peers. Peers execute their work units and return results to the centralised server. Examples of such systems include SETI@home [128] for searching for extraterrestrial intelligence by analysing radio telescope data and folding@home [110] for studying diseases by modelling the protein-folding process.

P2P systems have been also used for distributed data storage by exploiting disk space of participating peers. When a file is stored, it is divided into smaller chunks and each chunk is stored at some of the participating peers. To retrieve the file, all of its chunks are located and downloaded from peers. A DHT overlay is typically used to store and locate data chunks. Examples of P2P storage systems include OceanStore [63], PAST [28] and CFS [24].

Furthermore, P2P systems have been used for delivery of large-volume content to a large number of users by exploiting network bandwidth of participating peers [69, 73, 5]. In these systems, a peer may download content from other peers as well as upload content to other peers. This significantly reduces bandwidth requirements of a provider of content. In the following sections, we discuss P2P approaches to file-sharing, live streaming, and on-demand streaming.

2.2 P2P File-Sharing

In P2P file-sharing systems, files are located at peers and exchanged through direct connections between peers rather than through a centralised server. The importance of P2P file-sharing systems is reflected by the fact that they constitute a large fraction of the total Internet traffic. It is estimated that P2P file-sharing systems constitute

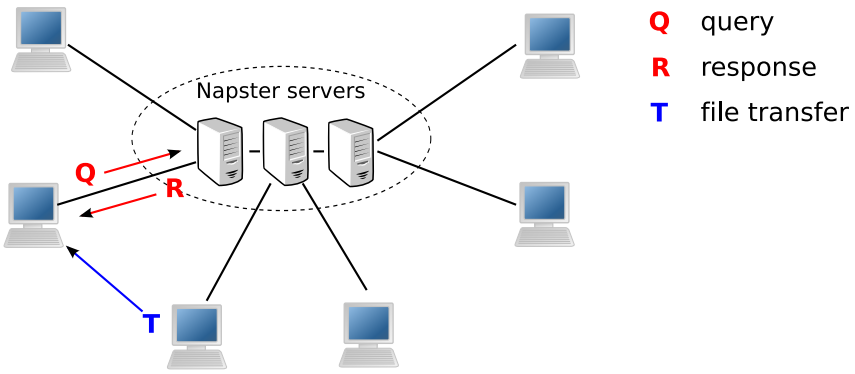


Figure 2.1: Napster architecture.

between 48% and 80% of the total Internet traffic, depending on the region of the world [54].

We distinguish three classes of P2P file-sharing systems. The first class includes early file-sharing systems that use centralised file discovery and transmit files directly between peers, but do not use a P2P overlay to improve performance of downloads. The second class includes systems that use a system-wide P2P overlay for decentralised file discovery, but do not use a P2P overlay to improve performance of downloads. The third class includes systems that focus on the performance of file downloads and, for that reason, form a P2P overlay for each file that is distributed.

2.2.1 Centralised

Napster [105] has been the first widely-used P2P file-sharing system. Figure 2.1 depicts its architecture. Napster uses a large number of dedicated central servers that maintain an index of the files shared by peers. A peer queries one of the central servers to obtain the set of peers that possess the requested file. The peer initiates a file transfer directly with any peer selected from the set.

2.2.2 System-Wide P2P Overlay

Gnutella [100] is a P2P file-sharing system that uses an unstructured P2P overlay for decentralised file discovery. Figure 2.2 depicts the architecture of early versions of Gnutella. File discovery works by flooding the overlay. To locate a file, a peer sends a query to all its neighbours. On receiving a query, a peer checks if it has the requested file. If so, it sends a response back to the query originator. In early versions of Gnutella, the response is sent back along the path that the query arrived. To improve performance, later versions of Gnutella send the response directly to the query originator. Irrespectively of the query match, a peer that received a query continues to flood the overlay by forwarding the query to all of its neighbours. The scope of flooding is only controlled by the Time-To-Live (TTL) parameter of the query, which is decreased with each hop and when it reaches zero, the query message is dropped. Once a peer locates a peer that has the requested file, it downloads the file directly from this peer.

In early versions of Gnutella, peers use ping and pong messages to discover new neighbours. Ping and pong messages behave similarly to query and query response messages. A peer periodically sends a ping message, which floods the overlay like the query message. Any peer that receives a ping message, sends a pong message back towards the originator of the ping message and also forwards the ping message to all its neighbours. Periodically, each peer connects to new peers discovered with ping/pong messages.

The flooding mechanism used by Gnutella has been shown to be inherently unscalable [101]. Flooding may also saturate connections of low-performance peers, resulting in queries being lost and such peers being unable to upload or download any files. To address these problems, more recent versions of Gnutella and other popular file-sharing systems, such as FastTrack/Kazaa [68], are based on super-peer architectures, as depicted in Figure 2.3. In these systems, the highest capacity peers are promoted and

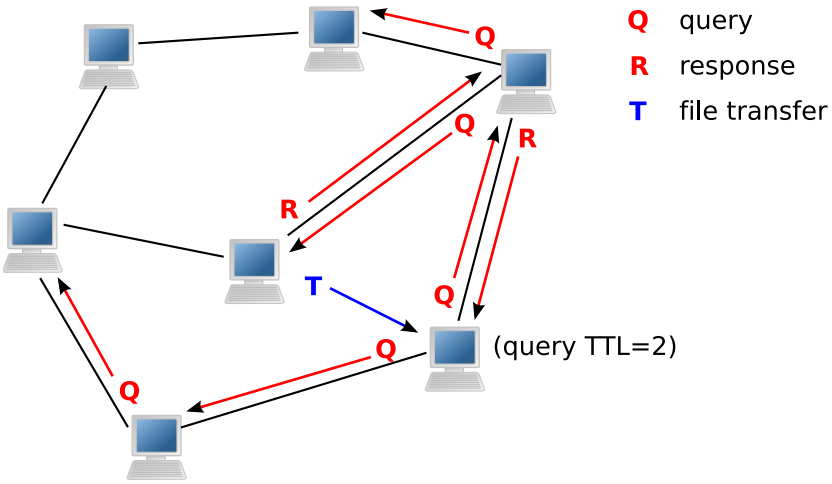


Figure 2.2: Gnutella architecture.

become *super-peers* (sometimes also called ultrapeers). Super-peers are interconnected with each other and form the core of the P2P overlay, where search queries are routed. Super-peers accept connections from ordinary peers and store indices of their files, forward their search queries to other super-peers and reply to search queries on their behalf. Thus, search queries flood only the core of the overlay that consists of high-capacity peers able to support higher network traffic. This significantly improves performance and scalability of such systems and has attracted further research attention [131, 81].

2.2.3 File-Level P2P Overlay

BitTorrent [22] is currently one of the most popular P2P file-sharing systems, estimated to generate somewhere between 50% and 75% of all P2P traffic on the Internet [54]. In contrast to many other P2P file-sharing systems, the BitTorrent protocol does not provide any file searching facility. Instead, the focus of BitTorrent and its strength lies in an efficient P2P file download.

The goal of BitTorrent is to enable distribution of a large file to many peers. The

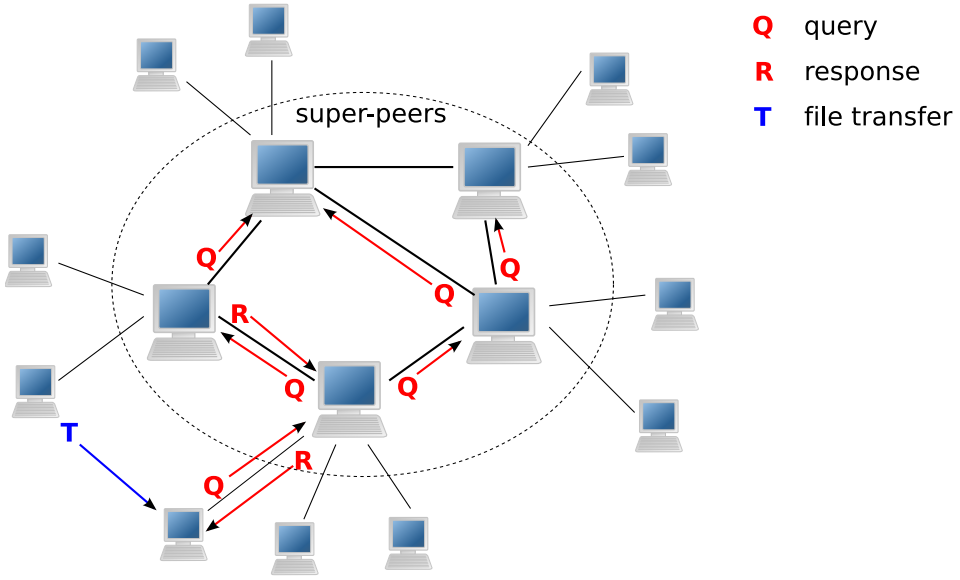


Figure 2.3: Gnutella super-peer architecture.

basic idea is to split the file into smaller equal-sized *chunks*, typically of a few hundreds kilobytes each, and enable peers to download chunks from multiple neighbouring peers in parallel. For each file available to download, there is a P2P overlay consisting of all peers participating in the distribution of this file and a central component, called a *tracker*, that keeps track of these peers. The tracker receives updates from peers periodically and when peers join or leave the overlay.

The overlay consists of peers that are either *seeders* or *leachers*. Seeders are peers that have a complete copy of the file and only upload chunks. Leachers are peers that have an incomplete copy of the file and that both download missing chunks and upload chunks, which they already have, to other peers. When a new peer joins the system, it connects to several dozen random peers, obtained from the tracker, that become its neighbours. The peer may later obtain additional neighbours if their number drops below some threshold due to peer churn. Peers attempt to download missing chunks from as many neighbours in parallel as they can. A peer uses a *local rarest first* policy to select chunks to download, meaning that the chunk with the fewest replicas among

neighbours is selected. The goal of this scheduling policy is to replicate all chunks evenly in the overlay in order to enable mutual exchanges of chunks between peers.

Figure 2.4 illustrates an example of chunks being exchanged by neighbouring peers. It shows one seeder, which has a complete copy of the file, and three leacher peers A, B and C. The tracker is not presented. In this example, peer A is connected to the seeder and peer C, peer B is connected to the seeder and peer C, and peer C is connected to peer A and B. Available chunks of each peer are represented as grey boxes. Unavailable chunks are represented as white boxes. The seeder has all chunks. The figure shows that each leacher downloads chunks from two neighbours in parallel. This is possible because leachers have a different set of chunks. If leachers had the same set of chunks, they could download new chunks from the seeder only.

BitTorrent uses a *tit-for-tat* policy to prevent free-riders, which are peers that download but do not upload content. Tit-for-tat is used by peers to preferentially upload chunks to peers that provide the highest download rate. Each peer limits the number of parallel uploads to a small number of neighbours, typically 5. The selected neighbours to which the peer uploads are referred to as *unchoked* and the remaining neighbours are referred to as *choked*. Every 10 seconds, the peer reevaluates each neighbour and *unchokes* neighbours that provide the highest download rate and chokes the remaining neighbours. Thus, peers that do not upload are punished by not being able to download.

Finally, BitTorrent uses an *optimistic unchoke* mechanism by which, in addition to normal unchokes, a peer periodically (every 30 seconds) unchokes a randomly chosen neighbour regardless of the download rate received from that neighbour. This mechanism enables peers to initiate chunk exchanges with new neighbours that may potentially provide a better download rate. It also enables newly joined peers to obtain their very first chunks so that they can begin exchanging these chunks for other chunks.

The performance of BitTorrent has been confirmed by many analytical [97, 78] and

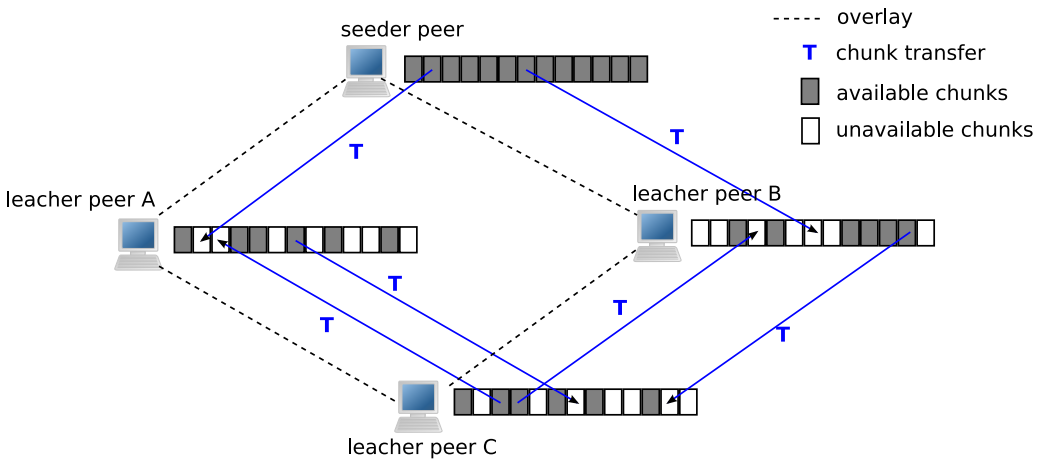


Figure 2.4: BitTorrent chunk distribution.

measurement-based [55, 92] studies. BitTorrent has also generated further research in mesh-based content delivery systems [30, 44, 10, 61, 39].

2.3 P2P Live Streaming

Media streaming can be live or on-demand. In live streaming, a live media content is disseminated to all viewers in real-time. The playback of all viewers is synchronised, meaning that all viewers watch the same content at the same time. In contrast, on-demand streaming is used for pre-recorded content and allows asynchronous playback, where different viewers may watch different parts of the same archived media file. In the following subsections we distinguish approaches based on a single multicast tree, multiple multicast trees, and a mesh overlay.

2.3.1 Single Multicast Tree

Figure 2.5 illustrates the approach in which peers are organised into a tree-structured overlay [20, 57, 18] with the root at the content provider, called the *transmitter*. The tree structure defines routing decisions – a peer receives a media stream from its parent

and forwards it to all its children. A peer that has children is called an *interior node*, whereas a peer with no children is called a *leaf node*. The terms peer and node are used interchangeably. In the figure, peers 0, 1, 2, 3 and 5 are interior nodes and peers 6, 7, 8, 4, 9 and 10 are leaf nodes. Two considerations important for tree construction include the height of the tree and the out-degree of interior nodes in the tree. The height of the tree is defined as the length of the longest downward path from the root to a leaf node. In the figure, the height of the tree is equal to 3. The out-degree of a node is defined as the number of children of that node. The height of the tree has an impact on the delay in the playback experienced by peers. Nodes closer to the root receive the media stream before nodes that are further from the root. For that reason, it is desirable to minimise the height of the tree. This can be achieved by increasing the out-degree of interior nodes. However, the out-degree of a node is constrained by the node's uplink bandwidth, which needs to be sufficient to upload the media stream at the rate of this stream to all children.

Approaches based on a single multicast tree have the following drawbacks. First, they do not use the uplink bandwidth of a large fraction of nodes in the tree. Leaf nodes do not have children, and thus do not upload any content. Second, the download rate of a node is limited by the minimum bandwidth on the path from the transmitter to that node. Any data loss in an upper level of the tree reduces the download rate of nodes lower in the tree. Finally, tree structures offer poor resilience to peer churn. Departure of an interior node in the tree results in the media stream being lost at all its descendants until the tree structure of the overlay is reconstructed.

Tree reconstruction is complex and incurs much overhead for two reasons. First, the out-degree constraint of nodes must be respected to avoid overloading nodes. Second, loops in the tree must be avoided. A loop is formed when a node becomes its own descendant in the tree. When a loop forms, nodes in the loop cease to receive recent content. For these reasons, multicast tree construction and maintenance is a challenging

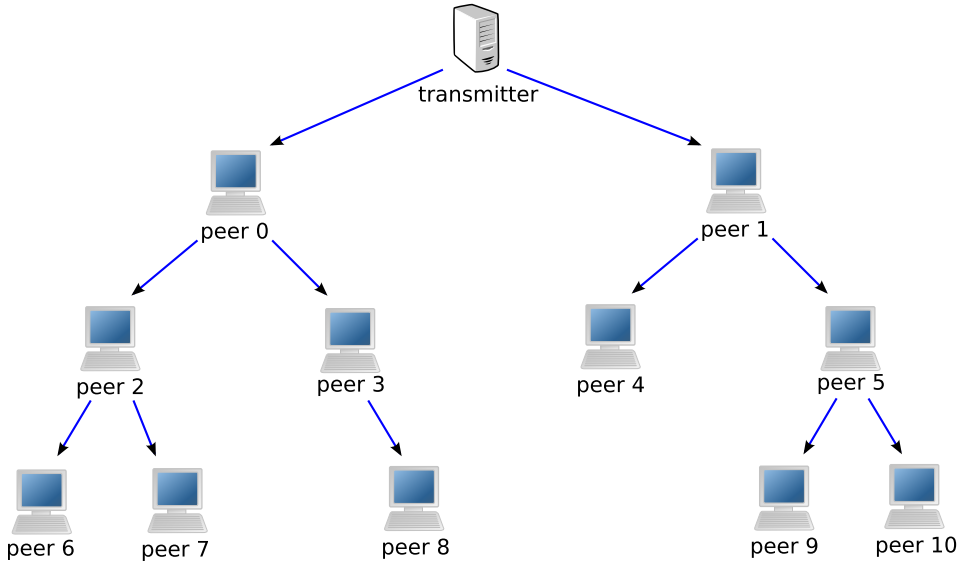


Figure 2.5: A single multicast tree overlay.

task that has attracted much research effort [133].

2.3.2 Multiple Multicast Trees

To address the problem of unused uplink bandwidth of leaf nodes, approaches based on multiple multicast trees have been proposed [88, 17, 121]. In these approaches, the transmitter splits the media stream into multiple disjoint substreams and sends each one using a distinct multicast tree. In order to receive all substreams and reconstruct the original stream, a peer joins all trees. Multicast trees are typically built so that each peer is an interior node in at most one tree and a leaf node in the remaining trees. The number of children of a node is limited by its available uplink bandwidth.

Figure 2.6 shows an example of live streaming based on multiple multicast trees with 2 substreams and 7 peers. The transmitter splits the stream into 2 substreams and pushes them into left and right multicast trees. Peers 0, 1, and 2 are interior nodes in the left tree and leaf nodes in the right tree. Peers 3, 4, and 5 are interior nodes in the right tree and leaf nodes in the left tree. Peer 6 is the only peer that is a leaf node

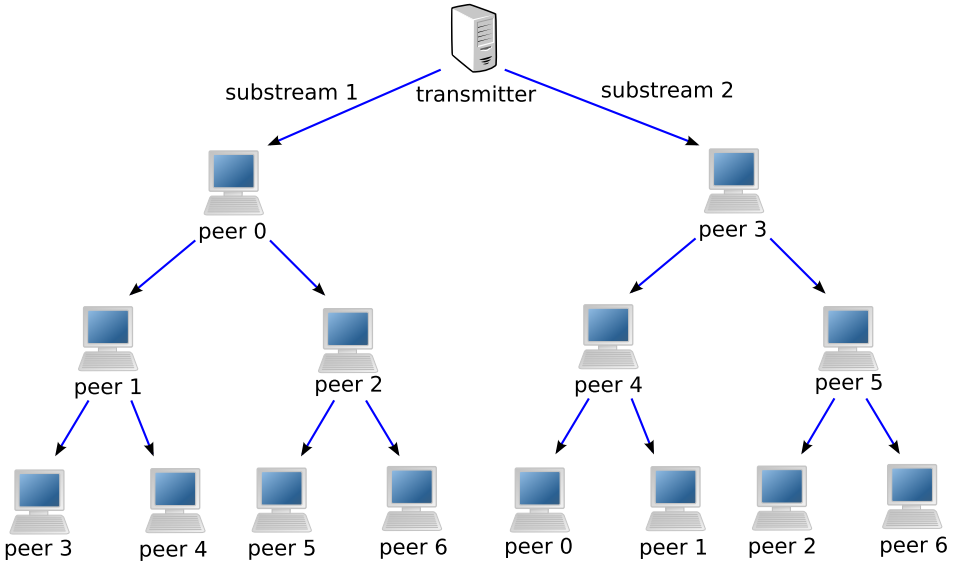


Figure 2.6: Multiple-tree-based approach for 2 multicast trees.

in both trees and does not contribute its uplink bandwidth.

The approach based on multiple multicast trees uses the uplink bandwidth of most of participating peers by placing them as interior nodes in one of the trees. However, maintaining such complex multicast tree structures and continuously adapting them to peer churn and to changing uplink bandwidth of peers presents a significant overhead that limits the efficiency of this approach [76].

2.3.3 Mesh Overlay

Mesh overlays offer an approach to P2P live streaming that does not require building and maintaining multicast trees [89, 75, 135, 11, 12]. Mesh-based P2P live streaming has been inspired by the mesh-based approach to P2P file-sharing. However, in contrast to file-sharing systems, the transmitter in P2P live streaming systems does not have access to the entire content. Content is generated "live" and so the transmitter cannot split the whole content into chunks for distribution throughout the overlay. In order to leverage mesh-based delivery, live streaming requires a delay between the stream

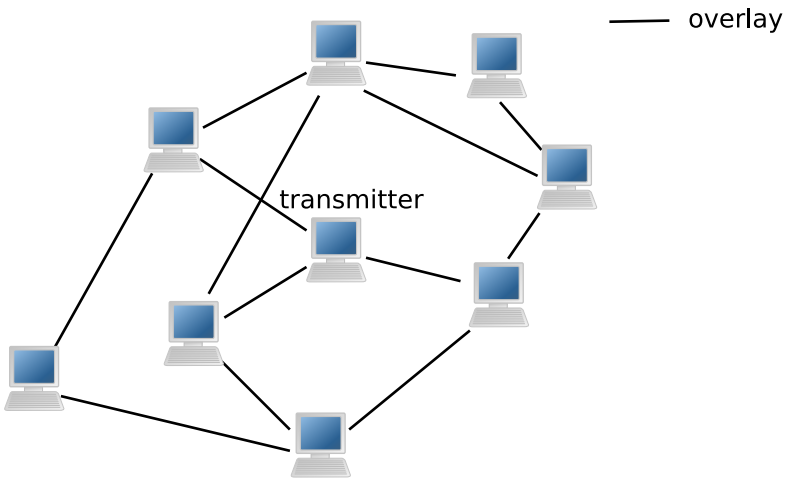


Figure 2.7: Mesh overlay.

creation time at the transmitter and the playback time at the viewer. The media stream produced within this delay is split into small consecutive chunks and distributed throughout the overlay similar to the way chunks of an entire file are distributed in mesh-based file-sharing systems.

A mesh overlay has no explicit structure and is typically formed by peers connecting to multiple other peers selected at random as illustrated in Figure 2.7. Peers inform their neighbours about chunks they download from other neighbours so that neighbours can request these chunks. Chunks downloaded by a peer before their playback time are stored in a sliding buffer and removed after their playback time. Chunks that do not arrive before their playback time result in playback interruptions.

Figure 2.8 shows an example of chunk exchanges in mesh-based P2P live streaming between three peers and a single transmitter. In this example, peer A is connected to the transmitter and peer C, peer B is connected to the transmitter and peer C, and peer C is connected to peer A and B. The transmitter has all chunks preceding the currently generated chunk. The figure shows how peers A, B and C maintain buffers that start at their current playback position and how they download chunks, which are

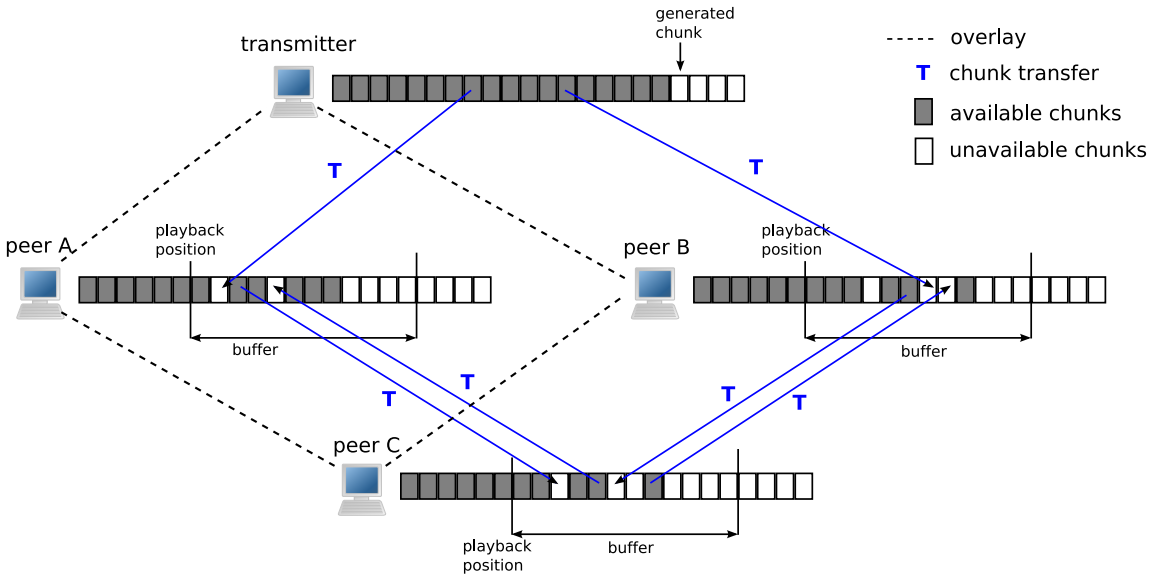


Figure 2.8: Mesh-based live streaming.

ahead of their playback position, from their neighbours.

In mesh-based approaches, a peer does not depend on any particular neighbour to download data chunks. When a neighbour of a peer fails or leaves the overlay, the peer can still download data chunks from its remaining neighbours. This is in contrast to approaches based on a single or multiple multicast trees, where a peer can download media exclusively from the parent that provides this particular media (sub)stream. Thus, the mesh-based approach is more resilient to peer churn and to fluctuations in the uplink bandwidth of peers. It also enables using the uplink bandwidth of all peers without the overhead of maintaining multiple tree structures.

Hybrid overlays, which combine multiple types of overlays, have also been proposed for P2P live streaming. Bullet [62] combines a single multicast tree with a mesh overlay. A peer receives a subset of chunks from its parent in the tree and the remaining chunks from its neighbours in the mesh overlay. This approach has been shown to use the bandwidth of peers more efficiently than approaches based on a single multicast tree [62], but less efficiently than mesh-based approaches [89].

2.4 P2P On-Demand Streaming

In on-demand streaming, viewers may watch any part of a media file at any time. On-demand streaming is often used for video files and thus it is often called Video On-Demand (VoD). Compared to file download, in VoD, a viewer does not need to wait until the whole media file is downloaded before she can start watching. This means that a viewer needs to download chunks of the media file approximately in sequential order, where each chunk is downloaded before its playback time. P2P techniques for efficient file download, such as those in BitTorrent, cannot be directly applied to VoD as they rely on non-sequential download of chunks. Non-sequential download of chunks maximises opportunities for chunk exchanges between peers.

Compared to live streaming, VoD allows viewers to watch pre-recorded media files whenever they want rather than at a specified time. However, this means that viewers may watch the same media file asynchronously and so they may download different parts of the same media file at any time. This hinders exchanges of content chunks between peers. In the following subsections, we introduce main approaches currently used for P2P on-demand streaming.

2.4.1 Patching

P2Cast [47] is a tree-based VoD system based on a *patching* technique that has been initially proposed to support VoD using IP multicast [53]. The general idea is to group peers that receive the same media file and that have a similar playback time into sessions. For each session, a multicast tree is formed through which peers receive the same portion of the media file.

Here, we consider a scenario with a single media file and arriving peers beginning their playback from the beginning of the media file. A peer arriving to the system joins the most recent session if the difference between its arrival time and the arrival time

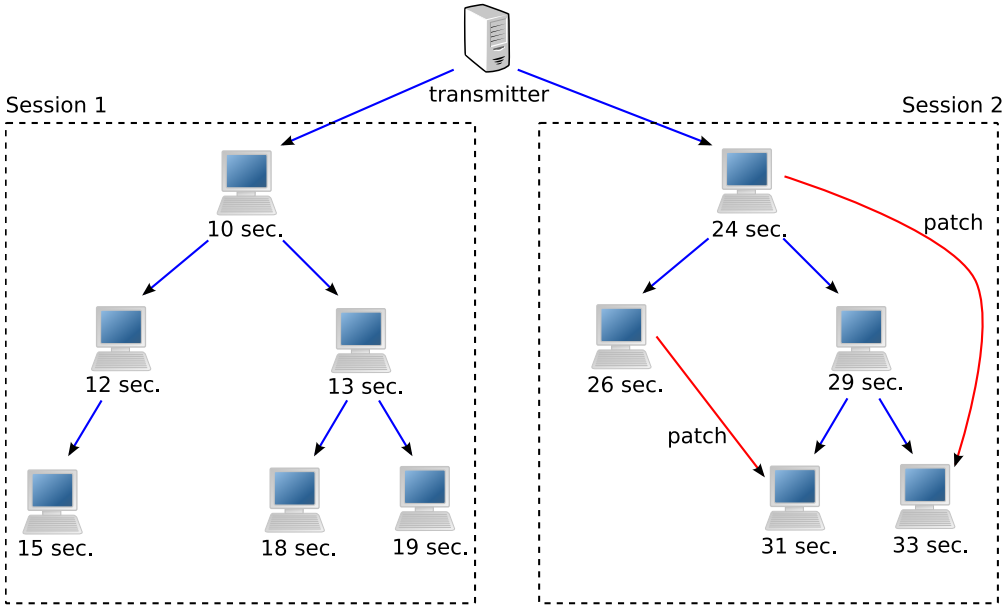


Figure 2.9: Snapshot of an overlay formed with the VoD patching technique. The current time is 34 seconds. The arrival time of peers to the system is indicated beneath peers. The threshold for each session is 10 seconds. Peers with arrival time of 31 and 33 seconds are still downloading patches.

of the first peer in this session is within a predefined threshold. Otherwise, it creates a new session and a new multicast tree. The arriving peer then starts receiving the media stream from the existing or newly created multicast tree. If the peer joined an existing session, it additionally needs to obtain the missing portion of beginning of the media file. This missing portion is called a *patch* and can be downloaded directly from the server or from the cache of any peer that has already downloaded it.

Figure 2.9 shows a snapshot of an overlay formed with the patching technique. The snapshot is taken at time 34 seconds. It shows a single transmitter and 11 peers that arrived at different time, which is indicated beneath each peer. The first peer arrived at time 10 seconds and created the first session and the first multicast tree. The subsequent 5 peers arrived within the 10 seconds threshold and therefore joined the same session and the same multicast tree. The peer that arrived at time 24 seconds

was outside the threshold and thus created a new session and a new multicast tree. The last two peers arrived at time 31 and 33 seconds, and therefore they joined session 2. The figure shows these two peers currently downloading patches that consist of, respectively, 7 and 9 seconds of the beginning of the media file.

2.4.2 Cache-and-Relay

In a *cache-and-relay* technique for P2P VoD [48, 23, 109], each peer stores in a cache the most recently downloaded portion of the media file. Peers download content from the cache of other peers that have a similar playback time. An example of this technique is illustrated in Figure 2.10. In this example, a number beneath peers indicates their current playback time. Each peer caches up to 10 minutes of the media file. A portion of the media file (in minutes) that is cached by each peer is indicated in square brackets. The overlay resembles a tree, however, peers may upload different portions of the cached media file to each of their children. The overlay is formed by each peer selecting a parent so that the cache of the parent covers the playback time of the peer. For example, the peer with the playback time at 36 minutes can download content from its parent with the playback time at 45 minutes. This is because the parent's current cache, which covers minutes from 35 to 45, contains content currently needed by the peer. The peers with playback time at 50 and 20 minutes can download content only from the transmitter as their playback time is not covered by the cache of any peer.

2.4.3 Mesh-based

Mesh-based approaches for P2P VoD [125, 6, 38] are based on mesh-based approaches for file-sharing and live streaming. Figure 2.11 shows the general idea of mesh-based VoD streaming. The media file is split into chunks. Peers download chunks from other peers and cache downloaded chunks. However, in contrast to both file-sharing and live streaming, peers in VoD are asynchronous to each other and interested in

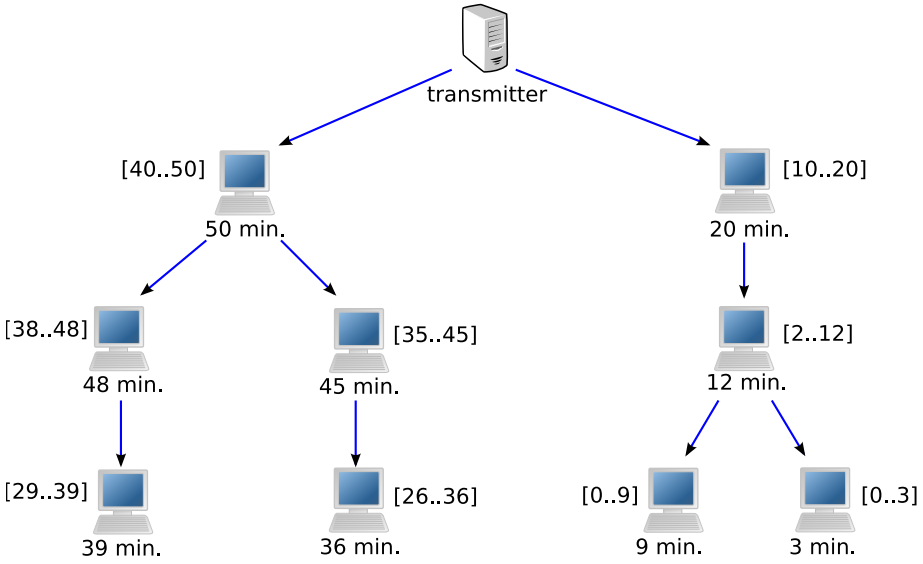


Figure 2.10: Snapshot of an overlay formed with the VoD cache-and-relay technique. The current playback time of peers is indicated beneath peers. Each peer caches up to 10 minutes of the media file. A portion of the media file (in minutes) currently cached by each peer is indicated in square brackets.

different parts of the media file. In a mesh overlay, peers have random neighbours that may have distant playback times. To increase chances that downloaded chunks can be uploaded to neighbouring peers with more advanced playback, peers download chunks in a non-sequential order. However, to deliver continuous playback, peers should download chunks in roughly sequential order. This situation is depicted in the figure. The peer with less advanced playback has few chunks that can be uploaded to the peer with more advanced playback. Thus, its uplink bandwidth may not be efficiently utilised, which reduces the performance of content distribution. Therefore, the main challenge of mesh-based P2P VoD systems is to find a balance between two opposing goals: diversity of downloaded chunks for high performance of content distribution and sequential download of chunks for continuous media playback.

A method for selecting chunks to download in mesh-based P2P VoD systems is proposed in [125]. In this method, peers categorise missing chunks into a high-priority

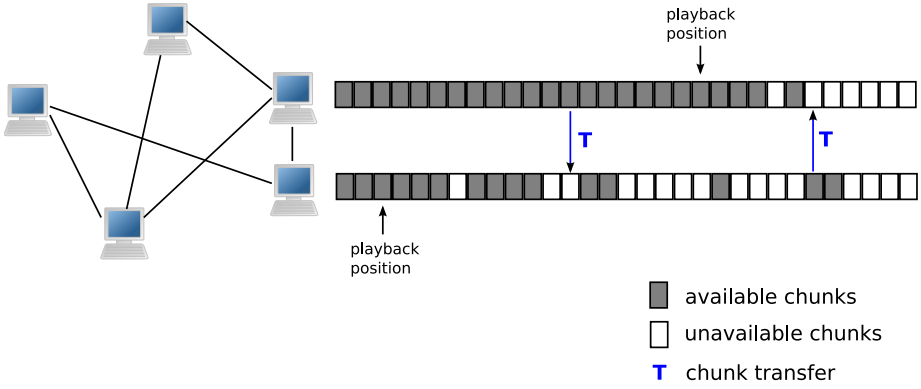


Figure 2.11: Example of a mesh-based P2P VoD technique.

set, which contains missing chunks that are close to their playback time, and a low-priority set, which contains the remaining missing chunks. To select a chunk to request from a neighbour, a peer first probabilistically selects one of these sets, and then randomly selects a chunk within the selected set. The high-priority set is selected with a probability p and the low-priority set with a probability $1-p$, where $0.5 < p \leq 1$. The value of p represents a tradeoff between continuity of playback at peers and performance of content distribution. By increasing p, the chances of downloading chunks before the playback time are increased. By decreasing p, the diversity of downloaded chunks is improved, which may result in a better performance of content distribution.

2.5 Network-Level Multicast

IP multicast [25] has been designed as a network efficient approach to one-to-many and many-to-many communication. An example of one-to-many communication is live media streaming from a content provider to many viewers. Examples of many-to-many communication include group video conferencing and online gaming, where each participant generates data that need to be sent to all other participants. IP multicast reduces network usage by enabling a sender to send each data packet only once, while

routers in the Internet forward this data packet to all receivers. It uses the notion of a *multicast group* that consists of all hosts receiving data in a particular multicast session. Currently, IP multicast does not support any means of access control. IP multicast allows any user on any host to create a group, receive data from any group and send data to any group. IP multicast is best-effort and unreliable, meaning that messages can be lost or delivered out-of-order. Each multicast group is identified by an IP address assigned from the class-D group of IP addresses [4]. There is no possibility of reserving multicast addresses or preventing applications from using the same multicast address.

To join a multicast group, a host contacts its network router using the Internet Group Management Protocol (IGMP) [15]. Network routers form and maintain a multicast spanning tree connecting all participants in the multicast group in order to efficiently disseminate data [127, 82, 1, 32, 65]. After a host joins a multicast group, it receives all data sent to this group, regardless of the sender.

2.5.1 Deployment

Although IP multicast significantly reduces the consumption of network resources in one-to-many and many-to-many communication, its practical deployment issues have prevented its wide-scale adoption. Its availability is currently limited mainly to academic institutions. Below, we briefly outline the issues related to the deployment of IP multicast. A comprehensive analysis of the reasons for the failure of wide-scale deployment is available in [27].

- IP multicast requires that routers maintain state for each multicast group they participate in. However, to achieve high performance, routers in the Internet backbone have a stateless architecture, dedicated to forwarding packets. IP multicast is expected to have a significant negative impact on the performance of routers that maintain a large number of multicast groups.

- IP multicast has security issues that may result in large-scale Denial of Service and flooding attacks by malicious users. Lack of access control in IP multicast allows any user to send any data to any group.

- IP multicast has no address allocation mechanism. Multicast traffic of different applications may merge together, causing conflicts between these applications. A scalable global allocation of unique multicast addresses is difficult to achieve, considering that applications, throughout the Internet, may need to frequently setup and release multicast groups.

- IP multicast is a best-effort service. Currently, despite much research effort [112, 2, 33], there is no effective and scalable higher-level protocol operating on the IP multicast layer to support reliable delivery and to provide error, flow and congestion control mechanisms.

- Finally, IP multicast requires changes to the network infrastructure. Most ISPs are reluctant to provide IP multicast support due to significant investment required as well as its scalability and security issues.

2.6 Discussion

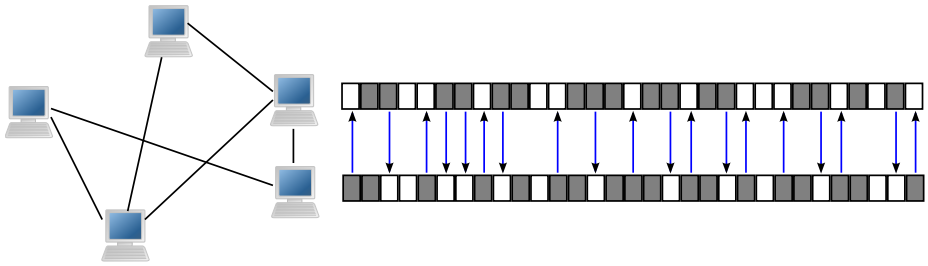
In this chapter, we introduced the main concepts of P2P systems. We surveyed the main P2P approaches for file-sharing, live streaming, and on-demand streaming and identified issues and challenges related to these approaches.

File-sharing, live streaming, and on-demand streaming have different characteristics that require distinct P2P approaches. However, an analogy between approaches for these three application domains can be drawn from the survey. In particular, an approach based on a mesh overlay has been proposed for each of these application domains. The reason for the popularity of mesh overlays is that they offer resilience to

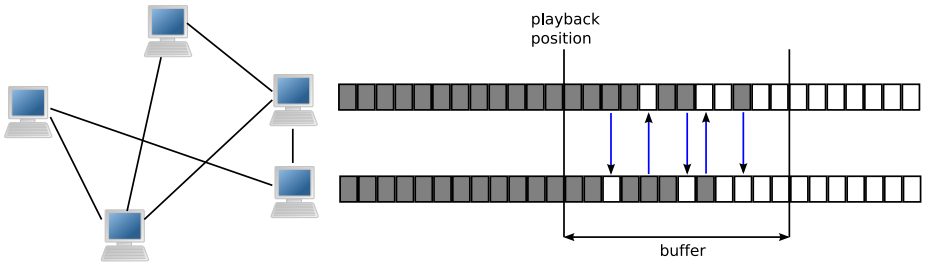
peer churn and low overhead of overlay maintenance. In mesh-based approaches, the content is split into small data chunks and peers download these data chunks from each other. However, different characteristics of file-sharing, live streaming, and on-demand streaming require that different methods are used to schedule downloads of these data chunks by peers.

In file-sharing systems, the order of downloaded chunks is not important as the goal is to download a complete file. Thus, peers download chunks in a non-sequential (fairly random) order to maximise the diversity of possessed chunks and thereby to maximise opportunities for chunk exchanges. Figure 2.12[1](a) illustrates the large number of possible chunk exchanges between two peers with random chunks. Live streaming is similar to file download in that peers are interested in the same content, and so any two peers may exchange chunks with each other. Figure 2.12(b) illustrates the possible chunk exchanges between two peers in the mesh-based live streaming approach that utilises peer buffers. Chunks within the buffer may be downloaded in a non-sequential order, which enables mutual exchanges of chunks between peers. However, contrary to both file-sharing and live streaming, peers in VoD are asynchronous to each other and interested in different parts of the media file. As illustrated in Figure 2.12(c), when two peers have different playback positions, bidirectional chunk transfers may not be possible. This poses difficulties in the dissemination of content in mesh-based VoD approaches. Nevertheless, the advantages of mesh overlays in terms of their resilience to peer churn and low overhead of overlay maintenance, attract much research interest in mesh-based approaches to VoD [125, 6, 38]. In the case of live streaming, mesh-based approaches have been shown to exhibit superior performance compared to alternative approaches [76]. Therefore, the work in this thesis focuses on mesh-based approaches.

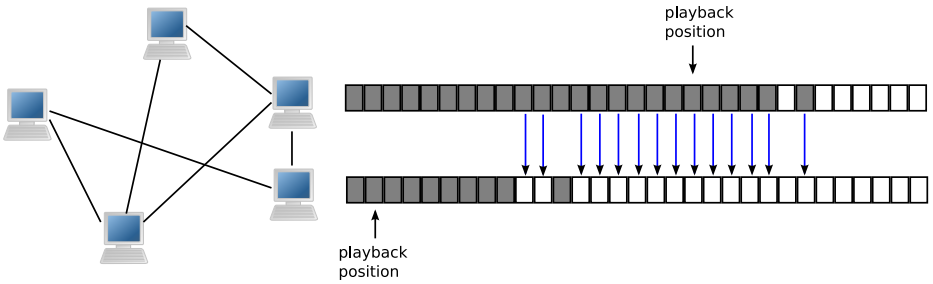
[1]Reproduced from [38]



(a) mesh-based file download



(b) mesh-based live streaming



(c) mesh-based on-demand streaming

Figure 2.12: Possible chunk transfers between two peers in different types of mesh-based P2P applications. An arrow indicates a possible transfer of a chunk between the peers.

Chapter 3

Adaptable P2P Live Streaming

P2P systems face challenges related to their high heterogeneity and dynamism. The heterogeneity comes from differences both in the amount of resources available at peers and in the quality of network connections between peers. The dynamism comes from transient population of peers, changing amount of available peer resources as well as changing quality of network connections. To accommodate heterogeneity and dynamism, P2P systems employ various adaptation methods. In this chapter, we review the state-of-the-art in adaptable P2P live streaming systems.

In Section 3.1, we present our criteria for selecting systems for the review and discuss various goals of adaptation in P2P live streaming systems. In Section 3.2, we introduce techniques for media stream coding, which are used in some of the reviewed systems. Sections 3.3, 3.4, 3.5, and 3.6 review and analyse each of the selected systems in the context of the presented adaptation goals. Finally, Section 3.7 discusses shortcomings in the state-of-the-art.

3.1 System Selection Criteria

The criteria to select systems for our review are based on our intuition of what is important from the perspective of a viewer in P2P live media streaming systems. In our opinion, the most important for a viewer is to receive uninterrupted playback at the maximum quality. As peer churn is often the main reason for playback interruptions at viewers, resilience to peer churn is the main criterion to select systems for our review. In live transmissions, peer churn may take an extreme form, called *flash crowd*, where a large number of viewers join or leave the system at approximately the same time. Flash crowds often coincide with the beginning and the end of live transmissions. Dealing with peer churn and flash crowds is particularly difficult, and often impractical, when the operation of a system relies on a tree-based P2P overlay structure. Hence, this review omits systems based on a single or multiple multicast tree overlay as they demonstrate poor resilience to peer churn and flash crowds [76]. In particular, we do not cover single-tree-based systems such as ESM [20], Overcast [57], ZIGZAG [118], MULTI+ [40] and Scribe [18]. We do not cover multiple-tree-based systems such as CoopNet [88] and SplitStream [17] that need to maintain even more complex overlay structures than single-tree-based systems. An exception to this is Chunkyspread [121] that is based on multiple multicast trees, however, its P2P overlay is highly dynamic and highly resilient to peer churn.

Our review covers Chunkyspread, Chainsaw [89], CoolStreaming [135] and PRIME [75] as they are resilient to peer churn and also they aim at improving the quality of playback delivered to viewers. These systems are reviewed and analysed in the context of their:

- **Resilience to peer churn.** We analyse how systems accommodate arrival and departure of peers to provide continuous media playback.
- **Adaptation to heterogeneous and dynamic bandwidth.** We analyse how

efficiently systems use the heterogeneous and dynamic network bandwidth of participating peers for media streaming. To maximise the download rate of peers and thereby to maximise the quality of peer playback, a system should adapt to utilise the entire heterogeneous uplink bandwidth of participating peers. Moreover, when the uplink bandwidth of peers changes, a system should adapt in a timely manner and without much overhead to avoid playback interruption at peers.

- **Adaptation of playback quality.** We discuss adaptation of the quality of playback at peers to their download rate. Download rates may vary among peers as the downlink bandwidth of some peers may reduce the download rate of these peers. Download rates may also vary over time as the uplink bandwidth available in the system may increase or decrease when new peers join the system, existing peers leave the system or peer uplink bandwidth changes. Thus, for the maximum quality of playback, a peer should continuously adapt its playback quality to its download rate.

Parallel to the research work, many commercial P2P live streaming systems have emerged in recent years. Examples of such systems include PPLive [93], SopCast [111], Zattoo [134], Feidian [29], PPStream [94], and TVants [119]. While these systems are proprietary, several studies bring insights into their characteristics and behaviour based on reverse engineering of the application code and network measurements [52, 51, 50]. However, these systems are not covered in our review as their design and algorithms remain largely unknown.

3.2 Media Stream Coding

In this section, we introduce techniques used for adapting the quality of playback at a viewer to the available bandwidth. Traditionally, this is achieved by the transmitter

offering multiple independent media streams encoded at different rates. Viewers manually select a media stream encoded at the rate just below their anticipated downlink bandwidth. However, the available bandwidth between the transmitter and a viewer may decrease, for instance, when other applications on the viewer's computer compete for bandwidth, when the transmitter becomes overloaded, or when congestion occurs in the Internet. When the available bandwidth drops below the rate of the selected media stream, playback may be interrupted. To handle this, the viewer needs to stop the current streaming session and initiate a new one at a lower rate and at a lower playback quality. *Layered coding* [42, 7] and *multiple description coding* [45, 95] are two types of techniques by which individual viewers may adapt the quality of playback to their available bandwidth without interrupting the playback. This is at the cost of loss of compression efficiency compared to media encoded at a single rate.

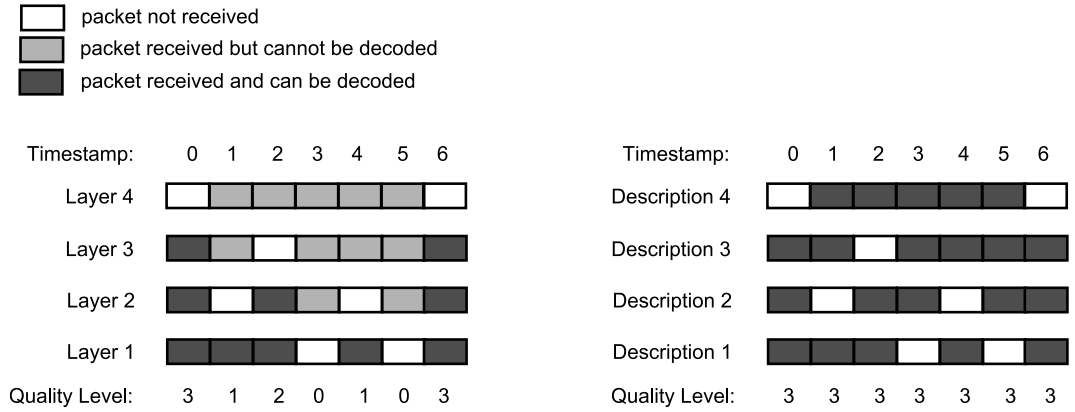
In Layered Coding (LC), also called embedded, progressive or scalable coding, the transmitter fragments a single media stream into M concurrent substreams ($M \geq 2$), called *layers*. Each layer has a rate lower than the rate of the original media stream, however, the exact rate depends on the LC method. The sum of the rate of all layers is, typically, higher than the rate of the original media stream due to the reduced compression efficiency of LC. Furthermore, layers are numbered and a viewer can decode any subset of the first k ($k = 1, ..., M$) concurrently received layers. In other words, layer $i + 1$ can be decoded only when the preceding i layers are correctly received. The quality of media playback corresponds to the number of decoded layers.

In Multiple Description Coding (MDC), the transmitter generates M concurrent substreams ($M \geq 2$) referred to as *descriptions*. Similar to LC, each description has a rate lower than the rate of the original media stream, while the sum of the rate of all descriptions is typically higher than the original rate due to the reduced compression efficiency of MDC. For playback, any subset of descriptions can be received. The distortion with respect to the original stream corresponds to the number of received

descriptions, i.e., the more descriptions received, the lower the distortion and the higher the quality of the reconstructed stream. This differs from LC in that in MDC every subset of descriptions can be decoded, whereas in LC layers are numbered and only subsets composed of a sequence of the first k ($k = 1, ..., M$) layers can be decoded.

Different MDC methods have been proposed [124]. For example, spatial polyphase downsampling produces M descriptions by storing i^{th} horizontal line of each frame in $(i \bmod M)^{th}$ description and independently encoding each description. In this method, the pixel resolution differentiates levels of the quality of playback. When k descriptions are correctly received, the pixel resolution of playback is reduced to $(k/M)^{th}$ of the original resolution. Likewise, temporal polyphase downsampling produces M descriptions by storing i^{th} frame of the media stream in $(i \bmod M)^{th}$ description and independently encoding each description. In this method, the frame rate differentiates levels of the quality of playback. When k descriptions are correctly received, the frame rate of playback is reduced to $(k/M)^{th}$ of the original frame rate. Another method, MDC-FEC [96], applies Forward Error Correction (FEC) coding to media encoded with LC to produce multiple MDC descriptions. In this method, the difference between levels of quality of playback depends on the implementation of LC.

As illustrated in Figure 3.1, MDC offers higher resilience to loss of random data packets compared to LC. Packet loss may be caused by network congestion or peer departures. The figure presents layers/descriptions that consist of consecutive packets represented as boxes. White boxes represent packets that are not received (i.e., lost) by the viewer. Light grey boxes represent packets that are received by the viewer, but cannot be used for decoding. A packet in a layer k cannot be used for decoding with LC if any of the concurrent packets in layers $1, ..., k-1$ is not received. Dark grey boxes represent packets that are received by the viewer and can be used for decoding the media stream. The *level of playback quality* of a viewer at timestamp t is defined as the number of packets received by the viewer such that these packets have the same



(a) Layered Coding (b) Multiple Description Coding

Figure 3.1: Comparison of the level of playback quality when LC and MDC are used in the presence of undelivered data packets.

timestamp t, belong to distinct layers/descriptions and can be used for decoding. In the example, the same set of packets is received by the viewer in the case of LC and MDC. However, all received packets can be used for decoding with MDC, whereas fewer packets can be used for decoding with LC. In particular, LC can use all received packets with timestamp 0, but only 1 packet (out of 3 received) with timestamp 1. This is because a packet in layer 2 and timestamp 1 is not delivered, and so packets in layers above this layer cannot be decoded with LC.

3.3 Chunkyspread

3.3.1 Review

Chunkyspread [121] is a P2P live streaming system that uses multiple multicast trees to distribute a media stream from the transmitter to peers. Peers adapt the multicast trees with the goal to use the uplink bandwidth of all peers efficiently. The secondary goal of peers is to reduce the *stream reception delay* of peers, which is called *latency* in

Chunkyspread. The stream reception delay of a peer is the time it takes to propagate the media stream through the overlay from the transmitter to the peer.

The transmitter partitions the media stream into M distinct substreams, called *slices*, and transmits them over separate multicast trees. Each peer joins M multicast trees, meaning that it has M parents, one in each tree. In each tree, a peer may be an interior or a leaf node. This is in contrast to other multiple-tree-based systems, such as [88, 17], where a peer is an interior node in at most one tree.

To efficiently use the uplink bandwidth of peers, each peer specifies its *target* and *maximum* load. The load of a peer is expressed as the aggregate number of children of the peer in all trees. The load of a peer corresponds to the consumption of its uplink bandwidth as the peer needs to concurrently transmit one slice to each of its children. Based on the target load, each peer determines its Upper Load Threshold (ULT) and Lower Load Threshold (LLT). If the peer's current load is within the range of the LLT and the ULT, then it is considered *satisfied*. If the load is below the LLT or above the ULT, then the peer is considered, respectively, *underloaded* or *overloaded*. A peer also determines its maximum load that must not be exceeded. The load thresholds of a peer are illustrated in Figure 3.2. Peers aim to adapt multicast trees so that every peer is satisfied, by having its load between the LLT and the ULT. If a peer's load is below the LLT, other peers will attempt to become its children in one of the trees, thereby increasing its load. If a peer's load is above the ULT, its existing children will attempt to find new parents, thereby decreasing its load. Once peers' loads are within the LLT-ULT range, they will no longer aim to improve load, but rather aim to reduce the stream reception delay. To achieve this, a peer may replace a parent if this action reduces the stream reception delay and does not cause the load of the new and the old parent to fall outside the satisfactory range.

Peers in Chunkyspread maintain a random overlay that is used to build and adapt multiple multicast trees. In this random overlay, each peer is connected to a random

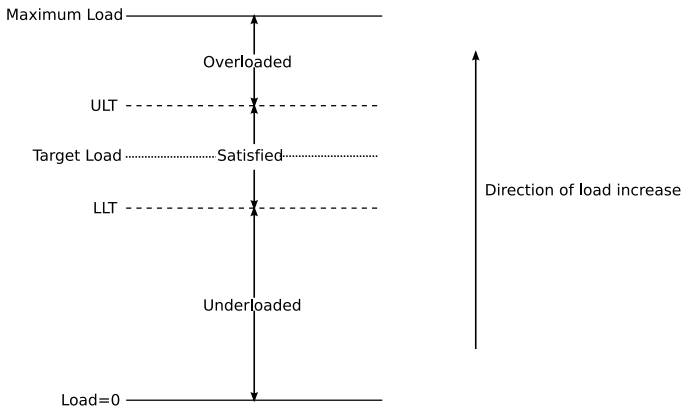


Figure 3.2: Load thresholds in Chunkyspread.

subset of all peers. Neighbouring peers exchange information about their current load, load thresholds as well as their stream reception delay and bloom filters, which we describe later. These information are used by peers to discover suitable parents for each tree. A peer is suitable as a new parent of another peer if this parent-child relationship satisfies a number of constraints. First, the maximum load of the parent should not be exceeded by adding a new child. Second, a loop in the tree should not be formed. A loop is formed in a tree when a peer becomes its own descendant in the tree. In order to detect loops, each data packet carries identifiers of the peers that forwarded this packet in the tree. To minimise the size of this information, bloom filters [129] are used. A bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. Peers collect bloom filters that they receive in each tree. To avoid loops, peers advertise these bloom filters to their random neighbours. A peer can become a child of a random neighbour without forming a loop in a tree, if the peer's identifier is not included in the bloom filter advertised by the neighbour for this tree. Nevertheless, a loop may form as many peers concurrently adapt the overlay. A peer detects a loop in a tree immediately when the peer receives from its parent in the tree a data packet that carries the peer's identifier. In such case, the peer replaces this parent with a random neighbour.

Parent-child relationships are periodically improved by peers searching for more suitable parents. If a peer has an overloaded parent and one of its random neighbours is underloaded, the peer replaces the overloaded parent with the underloaded neighbour as a new parent. This increases the load of the underloaded peer and reduces the load of the overloaded peer. When all neighbours of a peer are satisfied with respect to load, the peer looks for parent switches that can improve the stream reception delay. A peer estimates its relative distance from the transmitter in each tree by comparing the relative delay at which it receives packets for each slice. If a peer has a relatively large distance in one tree, it may replace its parent in this tree with a neighbour that has a relatively small distance in the same tree, if connection to the neighbour does not overload the neighbour.

3.3.2 Adaptability of Chunkyspread

Resilience to peer churn. Chunkyspread improves resilience to peer churn compared to other multiple-tree-based P2P live streaming systems by relaxing two constraints on the construction of multicast trees. First, each peer may be an interior node in multiple trees. This is in contrast to other multiple-tree-based systems that allow a peer to be an interior node in at most one tree so that the uplink bandwidth of every peer can be used to forward content. Second, the number of children of a peer is not fixed, but can fluctuate within the LLT-ULT interval. Relaxing these two constraints of multicast trees increases the number of peers suitable to become parents of a peer. As a consequence, peers can more easily find new peers suitable to replace existing parents that fail or leave the overlay.

Adaptation to heterogeneous and dynamic bandwidth. Chunkyspread adapts to the heterogeneous network bandwidth of peers. Its goal is to adapt the load of each peer to the peer's uplink bandwidth. However, a peer needs to specify its target

and maximum load, which depend on the peer's available uplink bandwidth. For that purpose, manual user input or bandwidth estimation tools [66, 117] may be used, but these methods are inaccurate and do not account for the dynamic nature of the network bandwidth.

A peer is considered satisfied if its load is close to its target load, which is below its maximum load. The margin between the target and the maximum load is used to accommodate variations in the number of children and the uplink bandwidth of the peer without the need to disconnect children of the peer. However, this margin corresponds to the amount of the uplink bandwidth that is unused at the peer. If the margin is large, much uplink bandwidth remains unused. In turn, a small margin may result in the load exceeding the maximum load. When the load exceeds the maximum load, multicast trees need to be reconstructed and descendants of the peer need to recover lost packets.

Adaptation of playback quality. Currently, Chunkyspread does not adapt the quality of playback. In [121], authors suggest that, in principle, it is possible to extend Chunkyspread so that the quality of playback adapts. This might be achieved using MDC. Descriptions created with MDC might be used as slices that are sent over distinct multicast trees. To downgrade the quality of playback, a peer might join only a subset of all multicast trees and thus receive only a subset of all slices. However, this requires changes in the construction of multicast trees and this has not been addressed.

3.4 Chainsaw

3.4.1 Review

Chainsaw [89] is one of the first P2P live streaming systems that uses a mesh overlay rather than multicast trees to distribute media streams. The mesh overlay is formed in a

random fashion by peers connecting to multiple peers selected at random. As discussed in Section 2.3.3, the transmitter splits the media stream into consecutive chunks of uniform length. The playback time at peers is delayed with respect to the stream creation time at the transmitter and peers maintain a buffer for storing chunks that are received before their playback time. To download chunks, peers request them from their neighbours. To enable chunk requests, neighbouring peers maintain local knowledge about the data chunks they possess by informing each other immediately when they receive a new chunk. A peer requests from a neighbour a chunk selected at random from those chunks that the peer is missing and the neighbour has available. The aim of the random selection of chunks is to increase the likelihood that neighbouring peers download different subsets of chunks and, as a consequence, can exchange chunks with each other. Peers upload and download chunks from multiple neighbours in parallel. Thus, a peer needs to keep track of what chunks it has requested from every neighbour to avoid requesting the same chunk from multiple neighbours. A *pipelining* technique [102] is used by which a peer may issue a request for a new chunk without waiting for a previous request to the same neighbour to be satisfied. Pipelining is used to eliminate the delay between the time when a peer completes transmission of a chunk to a neighbour and the time when it receives a subsequent request from the same neighbour. During this delay, no chunks are transmitted between these two peers and, consequently, their bandwidth may be unused. The number of requests pipelined to a single neighbour, however, is limited to ensure that requests are distributed across all neighbours. The ability to upload/download data chunks from multiple neighbours in parallel is one of the advantages of mesh-based systems. It enables to effectively use the downlink bandwidth of a peer as the uplink bandwidth of many neighbours is used. Moreover, it improves resilience to congestion. Congestion at a certain neighbour automatically causes less chunks being requested from this neighbour and more chunks being requested from remaining neighbours.

3.4.2 Adaptability of Chainsaw

Resilience to peer churn. Chainsaw improves resilience to peer churn compared to systems based on a single and multiple multicast trees. This is because in mesh-based systems a peer does not depend on any particular neighbour to download media content. A peer can download data chunks from any of its neighbours. When a neighbour of a peer fails or leaves the overlay, the peer can still download data chunks from its remaining neighbours. This is in contrast to approaches based on a single or multiple multicast trees, where a peer can download a media (sub)stream exclusively from the parent that provides this particular (sub)stream. When the parent fails, a peer needs to discover another suitable parent that satisfies various constraints imposed by multicast trees, such as an appropriate out-degree of the parent. Thus, the mesh-based approach proposed by Chainsaw is significantly more resilient to peer churn compared to tree-based approaches.

Adaptation to heterogeneous and dynamic bandwidth. In mesh-based approaches, all peers forward data chunks and so the uplink bandwidth of all peers may be used. This is in contrast to approaches based on a single multicast tree, where a large fraction of peers, the leaf nodes in the tree, do not upload content. Compared to approaches based on multiple multicast trees, mesh-based approaches avoid the complexity and overhead of maintaining multiple tree structures to utilise the uplink bandwidth of all peers. In addition, mesh-based systems adapt to variations in the uplink bandwidth of peers. When the available uplink bandwidth at a neighbour decreases, the peer automatically requests more chunks from remaining neighbours, thereby reducing traffic at the congested neighbour. In a tree-based system, when the available uplink bandwidth of a parent decreases, the parent may not be able to send a media (sub)stream at a sufficient rate to each of its children. In such case, some of its children need to search for new parents.

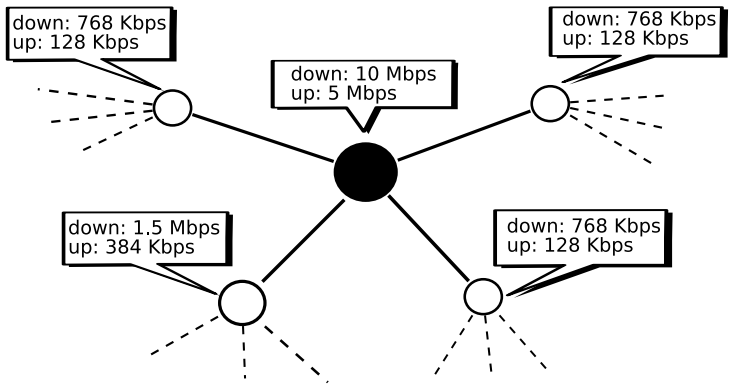


Figure 3.3: Example of possible inefficiencies in a random mesh overlay.

Chainsaw has been shown to achieve good utilisation of bandwidth in homogeneous Internet environments where all peers have the same and symmetric bandwidth [89]. However, this is not a realistic scenario as today's Internet consists of peers with heterogeneous and asymmetric uplink and downlink bandwidth [105]. In [11], we demonstrated that a random mesh overlay does not allow to use the entire uplink bandwidth of peers and causes download rates to be non-uniform among peers. Figure 3.3 shows an example of these inefficiencies of a random mesh overlay. The high capacity peer (black node) has four neighbours with low uplink bandwidth and hence its cumulative download rate is low, likely to be below the media stream rate. Thus, the black peer will likely experience playback interruptions. In addition, the black peer's uplink bandwidth is underutilised for two reasons. First, it cannot upload content to any single neighbour faster than it downloads content from neighbours. Second, the cumulative download capacity of its neighbours is lower than its uplink bandwidth, even if the neighbours were to download exclusively from the black peer. Therefore, the unused portion of the uplink bandwidth of the black peer reduces the streaming capacity of the system.

To deliver playback at the maximum quality to all peers, download rates need to be uniform among peers and need to maximise the usage of the uplink bandwidth of

peers. To provide such download rates, the P2P overlay needs to satisfy the following conditions:

- The overlay needs to allow to utilise the entire heterogeneous uplink bandwidth of all peers. A peer cannot upload content to any single neighbour at a rate higher than it downloads content. Thus, to utilise its entire uplink bandwidth, the number of peers to which it uploads should correspond to its uplink bandwidth. In contrast, a random mesh overlay assigns, on average, the same number of neighbours to all peers and thus a portion of the uplink bandwidth of high capacity peers may remain unused.
- The overlay needs to ensure that download rates are uniform among peers. A randomly formed mesh overlay may result in peers having neighbours with low uplink bandwidth only. As a consequence, the download rate of such peers may be below the rate of the media stream, resulting in playback interruptions.

Adaptation of playback quality. Chainsaw does not provide means to adapt the quality of playback at peers to their download rate. The media stream is encoded at a single rate for all peers and this rate does not adapt to the download rate of peers. Consequently, peers with the download rate below the rate of the media stream cannot deliver continuous playback. In turn, peers that may download at a rate higher than the media stream rate, deliver playback at a suboptimal quality.

3.5 CoolStreaming

3.5.1 Review

CoolStreaming [135] is a mesh-based P2P live streaming system. Similar to Chainsaw, the transmitter splits the media stream into chunks of uniform length. Each peer

maintains a buffer map that captures the availability of chunks in the buffer of the peer. In contrast to Chainsaw, peers do not notify neighbours immediately when they receive a new chunk. Instead, neighbouring peers periodically exchange their buffer maps. A peer uses buffer maps of neighbouring peers to periodically build schedules for fetching chunks from each neighbour. The scheduling algorithm aims to meet two constraints: the playback deadline for each chunk and the heterogeneous bandwidth from neighbours. The scheduling algorithm first calculates, for each chunk, the number of potential suppliers of this chunk (i.e., the number of neighbours that possess this chunk). Since a chunk with few potential suppliers is less likely to meet the playback deadline constraints, the algorithm determines the supplier of each chunk starting from those with only one potential supplier, then those with two, and so forth. Among multiple potential suppliers, the one with the highest bandwidth and sufficient time is selected. Once a peer generates schedules, it sends them to neighbouring peers, which then transmit selected chunks in the scheduled order. Further details and pseudo code of the scheduling algorithm can be found in [135].

The mesh overlay in CoolStreaming is initially random, formed by peers selecting neighbours at random. A gossip-based SCAMP protocol [37] is used by peers to periodically obtain random sets of peers. In contrast to Chainsaw, CoolStreaming employs an overlay adaptation algorithm to improve the mesh overlay. A peer periodically establishes a connection to a random peer in order to discover potentially better neighbours. To keep a constant number of neighbours, peers drop their currently worst neighbour in terms of its score. Peer i calculates a score for each neighbour j using function $max\{s_{i,j}, s_{j,i}\}$, where $s_{i,j}$ is the average number of chunks that peer i retrieved from peer j per unit of time. This adaptation tends to create connections between peers in which one of the peers can provide much useful data to another peer.

3.5.2 Adaptability of CoolStreaming

Resilience to peer churn. As discussed, CoolStreaming is a mesh-based pull-based system and, as such, provides high resilience to peer churn.

Adaptation to heterogeneous and dynamic bandwidth. CoolStreaming uses an algorithm to adapt a random mesh overlay so that two peers become neighbours if either of the peers can provide high upload rate to another. However, peers maintain a constant number of neighbours and, for this reason, CoolStreaming is unable to efficiently use the uplink bandwidth of high capacity peers. In live media streaming at a single rate, the maximum rate at which a peer can upload data to a single neighbour is limited by the rate of the media stream. Thus, if the number of neighbours is also limited, a peer with a sufficiently high capacity will be unable to fully use its uplink bandwidth. This results in a suboptimal use of the uplink bandwidth of high capacity peers.

CoolStreaming reacts slower to variations in the bandwidth of peers compared to Chainsaw. This is due to periodic scheduling of chunk downloads rather than immediate chunk requests. In Chainsaw, a peer distributes requests for chunks across all neighbours and issues new requests only when the old ones are satisfied. Thus, a decrease in the uplink bandwidth of a neighbour slows down the rate at which chunks are received from this neighbour as well as the rate at which chunks are requested from this neighbour. This results in more chunks being requested from the remaining neighbours of the peer. In contrast, when chunk downloads are scheduled periodically, the peer can react to a change in the uplink bandwidth of a neighbour only during the next scheduling round.

Adaptation of playback quality. Like Chainsaw, CoolStreaming does not adapt the quality of playback at peers.

3.6 PRIME

3.6.1 Review

PRIME [75] addresses performance issues of existing mesh-based P2P live streaming systems. Similar to our work in [11], the authors of PRIME identify that random mesh overlays inefficiently use bandwidth of peers, which is due to the reasons discussed in Section 3.4.2. Chunk dissemination in PRIME relies on a directed mesh overlay, where neighbours of each peer are divided into parents and children that, respectively, upload and download data chunks to/from the peer. In order to maximise the utilisation of both uplink and downlink bandwidth of all peers in the mesh overlay, PRIME uses the same ratio of bandwidth to peer degree for both uplink and downlink of all participating peers. More specifically, any two participating peers i and j satisfy the following condition:

$$bwpf = \frac{uplink_i}{outdegree_i} = \frac{downlink_j}{indegree_j}$$

where $outdegree_i$ is the number of children of peer i, $indegree_j$ is the number of parents of peer j, and $bwpf$, called bandwidth-per-flow, is the approximate bandwidth of each connection between any two neighbours in the mesh overlay. PRIME does not provide a decentralised algorithm for adapting the mesh overlay to meet these bandwidth-degree conditions. Instead, PRIME assumes that a centralised server, which has a global knowledge about all peers and their bandwidth, maintains the overlay.

The media stream is encoded with MDC. Each description is split into data chunks of uniform length. Peers control the quality of playback by requesting chunks in a desired number of distinct descriptions. Each peer periodically notifies its children about the data chunks it has received. A peer also, periodically, generates schedules for downloading chunks from each parent and sends each schedule to the respective parent. Peers send data chunks to their children according to the schedules received from them. The scheduling algorithm addresses the challenge of efficient distribution of chunks in

the mesh overlay. Each peer determines its target playback quality, which corresponds to the desired number of descriptions, by calculating the aggregate download rate from its parents. The scheduler gives priority to chunks with the highest timestamp, called diffusion chunks, in order to quickly distribute new chunks throughout the overlay. Subsequently, a peer schedules downloads of older chunks, called swarming chunks, taking into account the average data rate received from each parent and the desired number of descriptions. Further details of the scheduling algorithm can be found in [75].

3.6.2 Adaptability of PRIME

Resilience to peer churn. Content distribution in PRIME relies on a mesh overlay that is highly resilient to peer churn. However, PRIME does not provide any decentralised mechanism for the adaptation and maintenance of the mesh overlay. As is discussed below, a centralised approach does not scale.

Adaptation to heterogeneous and dynamic bandwidth. PRIME adapts a random mesh overlay so that the bandwidth of all peers is efficiently utilised for streaming. However, to achieve this, PRIME uses a centralised server. Moreover, it assumes that each peer estimates its uplink and downlink bandwidth, whereas such estimations are inaccurate using existing bandwidth estimations tools [66, 117]. Based on its bandwidth, a peer determines its appropriate in-degree (number of parents) and out-degree (number of children) and requests parents from the centralised server. The server ensures that each parent-child connection satisfies the out-degree constraints of the parent and the in-degree constraints of the child. To ensure this, the server needs a global knowledge about all participating peers and their current bandwidth. Such a centralised approach does not scale to a large population of peers that are prone to peer churn and variations in their bandwidth.

Similar to CoolStreaming, PRIME uses periodic scheduling of chunk downloads. For the reasons explained when describing adaptability of CoolStreaming, periodic scheduling reacts to bandwidth changes slower than immediate chunk requests.

Adaptation of playback quality. PRIME encodes the media stream into multiple descriptions and allows peers to individually determine the number of descriptions to request. To determine the number of descriptions to request, a peer calculates the aggregate download rate from its parents. Thus, a peer may increase the number of descriptions to request only when this download rate increases. However, it is not clear from the work presented in [75, 76] how this download rate increases. The download rate of a peer can increase only when the amount of chunks requested by the peer increases. In turn, the amount of requested chunks can increase only when the peer increases the number of descriptions to request. However, the peer can increase the number of descriptions to request only when its download rate increases. Following this logic, peers never increase the number of descriptions to request, and thus do not improve the quality of their playback.

Another inefficiency of PRIME results from the lack of cooperation between adaptation of the playback quality and adaptation of the overlay. For a high-quality playback, a peer needs to download chunks in multiple distinct descriptions. However, in PRIME, parents of a peer are selected randomly. Such random parents may have low downlink bandwidth, and thereby download few descriptions. As a consequence, random parents may not offer the number of disjoint descriptions sufficient for a high-quality playback at the peer.

3.7 Discussion

In this section, we discuss shortcomings in the reviewed systems and in tree-based approaches to P2P live media streaming. In particular, we discuss shortcomings in

their resilience to peer churn, adaptation to heterogeneous and dynamic bandwidth, and adaptation of the quality of playback. We also outline two other topics, locality-awareness and incentives for cooperation, that the work in this thesis does not address.

3.7.1 Resilience to Peer Churn

Tree-based systems. In single-tree-based systems [20, 57, 118, 18], an interior node in a tree provides the entire media stream to the whole subtree rooted at this node. Thus, departure of an interior node results in all nodes in its subtree ceasing to receive new data until the tree structure is reconstructed. Reconstruction of the tree is not straightforward as it needs to prevent loops in the tree and needs to respect the out-degree constraint of nodes, determined by their available uplink bandwidth.

In multiple-tree-based systems [88, 17], each multicast tree distributes a single media substream. If each peer is an interior node in at most one of the multicast trees, the overlay is said to be *interior-node-disjoint*. The interior-node-disjointness property guarantees that departure of a peer affects at most one multicast tree (one substream), in which the peer is an interior node. However, tree maintenance is difficult when interior-node-disjointness of the tree needs to be preserved. Some multiple-tree-based systems, such as CoopNet [88], rely on a centralised server, while other, such as SplitStream [17], rely on complex Distributed Hash Tables to manage the multiple multicast trees. To ease maintenance of multiple multicast trees, each peer in Chunkyspread may be an interior node in multiple trees. Moreover, Chunkyspread relaxes the out-degree constraint of nodes by allowing the out-degree to oscillate within some threshold. This makes it easier for peers to find suitable parents when they join the system or when their existing parents leave due to peer churn.

Mesh-based systems. Mesh-based live streaming systems typically provide better resilience to peer churn compared to tree-based systems. They rely on a mesh over-

lay and a pull-based approach to distributing media content. Mesh overlays support connections between any two peers in the overlay, which makes it easy for a peer to replace failed neighbours. In the pull-based approach, a peer explicitly requests missing data chunks from its neighbours. A peer does not depend on a specific peer to download data chunks. When a neighbour of a peer leaves the overlay or fails, the peer can download data chunks from its remaining neighbours.

3.7.2 Adaptation to Heterogeneous and Dynamic Bandwidth

P2P live streaming systems use the uplink bandwidth of participating peers for distributing media content. Therefore, the maximum rate of the media stream is directly related to how efficiently bandwidth of peers is utilised. A stream encoded at a single rate can be sent through the overlay at the maximum rate when the entire uplink bandwidth of all peers is utilised and when the download rate of all peers is the same. A theoretical study of overlay architectures that achieve this goal is presented in [106]. However, the study does not cover creation and maintenance of such overlays, which is not straightforward in dynamic environments that are prone to peer churn and variations in the uplink bandwidth of peers.

Tree-based systems. Single-tree-based systems use the uplink bandwidth of peers inefficiently due to a large fraction of peers, the leaf nodes, not uploading content. Work in [107] partially addresses this problem by adapting a multicast tree so that peers with high uplink bandwidth are interior nodes and peers with low uplink bandwidth are leaf nodes. This results in a lower amount of unused uplink bandwidth in the system. Placing peers with high uplink bandwidth near to the root of the tree has the additional benefit of reducing delays at which the stream is received by peers. This is because peers with high uplink bandwidth may have a large out-degree in the tree. A large out-degree of nodes near to the root reduces the height of the tree.

Bullet [62] addresses the problem of leaf nodes not uploading content by augmenting a single multicast tree with a mesh overlay. Peers receive a subset of chunks from their parents in the tree, while the remaining chunks are recovered from neighbours in the mesh overlay. Thus, peers that are leaf nodes in the multicast tree, may use their uplink bandwidth for uploading data chunks to neighbours in the mesh overlay. This approach has been shown to utilise the bandwidth of peers more efficiently than approaches based on a single multicast tree, but worse than pure mesh-based approaches [89].

In addition to the inefficient use of the uplink bandwidth, single-tree-based systems adapt poorly to variations in the uplink bandwidth of peers. An interior peer in a multicast tree needs to forward the entire media stream to each of its children. Thus, the number of children of the peer is determined by its available uplink bandwidth. When the uplink bandwidth of the peer decreases, some children may not be able to receive the entire media stream and thus may have to find a new parent.

Multiple-tree-based systems are able to efficiently use the uplink bandwidth of all peers by distributing media streams using multiple multicast trees. The uplink bandwidth of each peer can be used by placing the peer as an interior node in one of the multicast trees. However, similarly to a single multicast tree, multiple multicast trees adapt poorly to variations in the uplink bandwidth of peers. When the uplink bandwidth of a peer decreases, some children of the peer, possibly in different multicast trees, may need to find a new parent.

In Chunkyspread, each peer sets its target load, i.e., its desired number of children, below its maximum load, i.e., the maximum number of children that it can support. This helps to accommodate some oscillations in the number of children and the uplink bandwidth of a peer without the need to reconstruct the tree structures. However, the margin between the target and the maximum load results in a suboptimal use of the uplink bandwidth of peers.

Mesh-based systems. Mesh-based systems allow to use the uplink bandwidth of all peers for distributing media content, without the complexity of maintaining multiple multicast trees. They are resilient to changes in the network bandwidth, because a peer does not depend on a specific neighbouring peer to download data chunks. When the uplink bandwidth of a neighbour decreases, the peer can download data chunks from its remaining neighbours. However, our work in [11, 12] and PRIME show that a system based on a random mesh overlay, such as Chainsaw, is unable to utilise the entire heterogeneous uplink bandwidth of peers. Moreover, in such a system, download rates vary among peers. This is undesirable in streaming, where all peers with sufficient downlink bandwidth should receive media content at the same rate, maximising peer upload rates.

This thesis proposes the first fully decentralised algorithm for peers to adapt a random mesh overlay so that the upload rate of peers is maximised and the download rate of peers with sufficient downlink bandwidth is approximately the same.

3.7.3 Adaptation of Playback Quality

Adaptation of the overlay is necessary but not sufficient to deliver playback at the optimal quality to peers. This is for two reasons. First, the maximum rate at which a media stream can be sent through the overlay is unknown to the transmitter and fluctuates when peers join and leave the system and when their available bandwidth changes. Second, the downlink bandwidth of some peers may be below this maximum rate and thereby may not allow for continuous playback at these peers. These two problems may be addressed by encoding the media stream using MDC or LC, and by enabling peers to control the number of descriptions or layers to download.

Tree-based systems. Multiple-tree-based systems, such as SplitStream [17] and CoopNet [88], produce multiple descriptions using MDC and disseminate each description

using a distinct multicast tree. Individual peers adapt the quality of their playback to their download rate by subscribing to an appropriate number of multicast trees. The use of MDC also helps to prevent playback interruptions caused by packets being undelivered due to peer churn or varying peer bandwidth. When a data packet in a description is not received before its playback time, then, instead of interrupting the playback, corresponding received packets in other descriptions may be used for continuous playback at a reduced quality.

Mesh-based systems. Ensuring the appropriate quality of playback is more difficult in mesh-based systems than in multiple-tree-based systems. This is because a mesh overlay is constructed randomly and data chunks are disseminated in an unorganised, roughly random, fashion. To adapt the quality of playback, PRIME encodes a media stream using MDC and enables peers to determine the number of descriptions to download. However, for reasons described in Section 3.6.2, it is not clear how peers in PRIME determine the appropriate number of descriptions to download.

This thesis presents algorithms for peers to adapt their quality of playback to their download rate. The adaptation of the playback quality cooperates with the adaptation of the overlay so that a peer connects to neighbours that provide a sufficient amount of chunks for playback at the desired quality. This is in contrast to PRIME, where parents of a peer are selected randomly and thus may not offer a sufficient amount of chunks for playback at the desired quality.

To our knowledge, the work presented in this thesis is also the first to reduce the playback startup delay at peers. A peer that joins the overlay, initially downloads only a single description that corresponds to a basic playback quality and allows for a short startup delay. The basic playback quality is often sufficient for a viewer to decide whether to continue watching the transmission or to switch to a different one. The quality of media playback then gradually improves over time as the number of

descriptions to download is increased.

3.7.4 Other Adaptation Goals

Many other challenging problems remain to be addressed in the domain of P2P live streaming. Two topics that we consider important, but beyond the scope of this thesis are locality-awareness and incentives for cooperation.

Locality-awareness. P2P systems tend to increase the traffic of ISPs as peers first download content and then upload the content to other peers [103]. This generates roughly double the amount of traffic at the ISPs compared to client-server architectures, where most of the traffic flows in one direction. As a consequence, network traffic costs of ISPs often increase.

Locality-aware P2P systems exploit network proximity between peers to mitigate the impact of P2P on ISPs. They rely on the fact that the cost of transmitting content between peers located within an ISP's local network is significantly lower than the cost of transmitting content outside of the ISP's network. In these systems, content once downloaded from outside the ISP's local network may be shared by peers within the ISP's local network to reduce the network costs of the ISP.

Locality-awareness also reduces network traffic in the rest of the Internet as much of the traffic remains within local networks of ISPs. Finally, locality-awareness reduces the network distance of content transmissions, resulting in lower transmission delays and lower probability of packets being dropped by intermediate Internet routers.

Much work has been done on constructing locality-aware P2P multicast trees [40, 18, 17, 136, 70]. Locality-aware P2P multicast trees are build so that a parent and a child are close to each other in terms of network latency or their IP address. For instance, Scribe [18] and SplitStream [17] use the underlying Pastry DHT overlay [104] and its proximity-based routing mechanism to construct multiple locality-aware

multicast trees.

Contrary to P2P multicast trees, there has not been much research on improving proximity in mesh-based P2P streaming systems. Mesh-based systems typically form a mesh overlay in a random fashion and thus the overlay is not locality-aware. Rainbow [19] is a mesh-based system that addresses this problem. It uses a two-layer architecture. The lower layer consists of many clusters of peers with a root node for each cluster. Peers are assigned to a cluster so that peers in each cluster are nearby to each other in terms of network latency. The cluster root nodes form the upper layer, the backbone network. The backbone network and each cluster is constructed as a mesh overlay. Additional random connections are established between members of different clusters to improve data chunk dissemination and robustness to peer churn. Dissemination of data chunks in each mesh overlay is based on Chainsaw. Compared to Chainsaw, Rainbow improves network proximity of peer connections, however, it is less resilient to peer churn due to lower connectivity of peers. In particular, cluster root nodes are responsible for much of inter-cluster chunk transfers. Failure of the root node of a cluster may reduce the number of chunks available to peers in the cluster. Like Chainsaw, Rainbow does not adapt to efficiently utilise the heterogeneous bandwidth of peers.

The work presented in this thesis does not address the challenge of providing locality-awareness.

Incentives for cooperation. An important goal of all P2P systems is to ensure that participating peers cooperate with each other for the benefit of the whole system. In P2P live streaming systems, non-cooperative peers may refuse to forward media streams to other peers in order to save their own uplink bandwidth or peers may attempt to take advantage of the system by downloading at a higher rate than other participating peers. This type of non-cooperative behaviour may result in the "tragedy of the commons"

[49], when the correct functioning of the system becomes impossible. To address this, systems may use incentives for peers to cooperate. Incentives may be designed to ensure that peers upload approximately the same amount as they download. This can be achieved by peers favouring mutual content exchanges in which both parties send some content to each other at the same time [22, 80], or by using credit-based techniques, where a peer earns credits when it uploads content and spends credits for downloading content [83, 122]. However, such strict fairness may not be desirable in P2P live streaming systems as it may substantially reduce their performance. Fairness prevents the possibility that some peers are willing to contribute more bandwidth than they consume. It also rejects peers that cannot share fairly due to network configurations and firewalls. Therefore, different types of incentives have been proposed for P2P live streaming.

For instance, PULSE [90] is a P2P live streaming system designed with the goal to provide incentives for peers to upload content. However, it neither enforces that peers upload content nor limits their upload rate. It connects high capacity peers that upload much content and places them close to the transmitter. This results in high capacity peers benefiting in terms of a lower stream reception delay compared to the remaining peers. The whole system benefits in terms of improved stream reception performance as it prevents low capacity peers from distributing the most recent data chunks. Low capacity peers distributing the most recent data chunks might slow down or disrupt the content distribution process.

The work in this thesis assumes that all peers behave in a cooperative manner.

Chapter 4

MeshTV

In this chapter, we describe the mesh-based P2P live media streaming system, called MeshTV, that adapts the mesh overlay and the quality of playback so that peers deliver playback at the maximum quality and with a short startup delay.

In Section 4.1, we outline components and algorithms of MeshTV. Section 4.2 discusses dissemination of content in the MeshTV P2P overlay and proposes algorithms for peers to schedule downloads from other peers and for the transmitter to schedule uploads to peers. In Section 4.3, we derive algorithms for adapting the mesh overlay in order to deliver media content to peers at rates that are nearly uniform among peers and maximise the usage of peer uplink bandwidth. We summarise this chapter in Section 4.4.

4.1 MeshTV Overview

Main goals and characteristics of MeshTV are:

- **Scalability.** MeshTV uses only decentralised algorithms that scale to large populations of peers.

Peer	Transmitter
participates in the mesh overlay	
participates in the membership management protocol	
maintains a buffer for storing downloaded chunks	produces a media stream and splits it into chunks
maintains its own buffer map and the buffer map of each sender	maintains the buffer map of each receiver
selects chunks to download from different senders in parallel	selects chunks to upload to different receivers in parallel
adapts the mesh overlay	

Table 4.1: Comparison of responsibilities of peers and the transmitter.

- **Resilience.** MeshTV is based on a mesh overlay that is highly resilient to peer churn and variations in network bandwidth.
- **Optimal quality of playback.** MeshTV adapts the mesh overlay so that download rates are nearly uniform among peers and the upload rate of peers is maximised. Furthermore, MeshTV adapts the quality of playback of each peer to the download rate of the peer.
- **Short playback startup delay.** MeshTV adapts the quality of playback so that a peer joining the system delivers initially playback of a basic quality that allows for a short playback startup delay. The quality of playback gradually improves over time.

In MeshTV, we distinguish between a peer and a transmitter. A peer is run by a viewer and typically there is a large number of peers participating in each live transmission. In contrast, a transmitter is run by a content provider and there is a single transmitter for each live transmission. In each transmission, peers and the transmitter have different responsibilities that are outlined in Table 4.1 and discussed below.

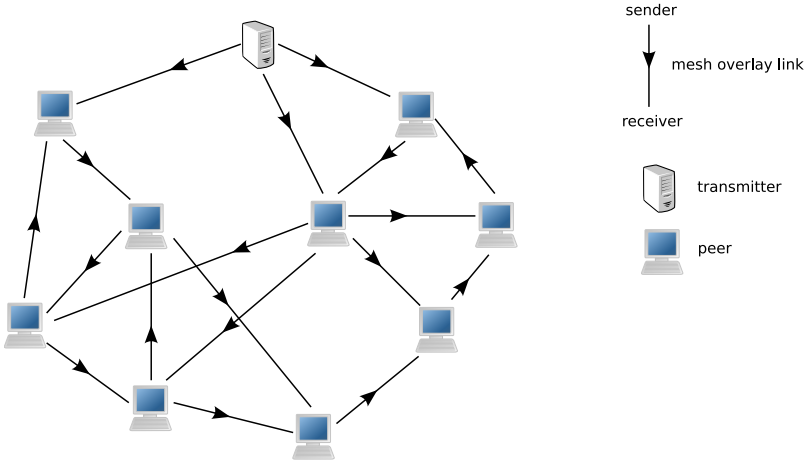


Figure 4.1: Example of the MeshTV overlay.

Mesh overlay. For each transmission, all peers and the transmitter form a mesh overlay that is illustrated in Figure 4.1. In this overlay, each peer and the transmitter is connected to multiple neighbours. Neighbours of a peer are divided into *senders* and *receivers*. Senders are neighbours from which the peer downloads content, while receivers are neighbours to which the peer uploads content. A peer has another peer in its receiver set precisely when the latter has the former in its sender set. The transmitter uploads, but does not download content, so it has no senders. Dividing neighbours into senders and receivers enables to control the use of the uplink bandwidth of a peer by increasing or decreasing the number of its receivers, without affecting its download rate from senders. This type of a mesh overlay can be seen as a *directed graph*, where connections with senders are incoming links of a peer and connections with receivers are outgoing links of a peer. *In-degree* of a peer defines its number of senders and *out-degree* defines its number of receivers.

To form the mesh overlay, peers joining the system select multiple random peers (or the transmitter) as their senders and connect to them. Peers connecting to senders automatically become receivers of these senders.

Membership management. A peer needs to discover other peers or the transmitter to join the system, to replace neighbours that fail or leave the system, or to adapt the P2P overlay. This problem is referred to as *membership management*. MeshTV uses a gossip-based membership management protocol from [126] that periodically provides each peer and the transmitter with a new random subset of addresses of participating peers.

Media stream. In MeshTV, the transmitter receives from a content provider a live media stream of a high quality and at a rate that is constant over time. The transmitter fragments this media stream into M concurrent media descriptions. To achieve this, it uses MDC-FEC [95] that is a popular MDC method used in many other P2P live streaming systems [88, 75]. MDC-FEC is selected because it produces each description at the same bitrate. Furthermore, the transmitter splits each description into consecutive data chunks of uniform length. Each data chunk is uniquely identified by the pair consisting of its *description number* and of its *timestamp*, where the timestamp identifies the position of the chunk within the media stream. These data chunks are then distributed in the mesh overlay.

Buffers and buffer maps. To accommodate dissemination of chunks throughout the overlay, the playback time at peers is delayed by approximately Δ seconds with respect to the time when the media stream is generated at the transmitter. Data chunks received by a peer before their playback time are stored in its internal *buffer*. Data chunks are removed from the buffer after their playback time.

Each peer maintains a *buffer map*, illustrated in Figure 4.2, that captures the availability of chunks in its internal buffer. In addition to its own buffer map, a peer maintains the buffer map of each of its senders to select chunks to download from the senders. In turn, the transmitter maintains the buffer map of each of its receivers to select chunks to upload to its receivers. By selecting chunks to upload, the transmitter

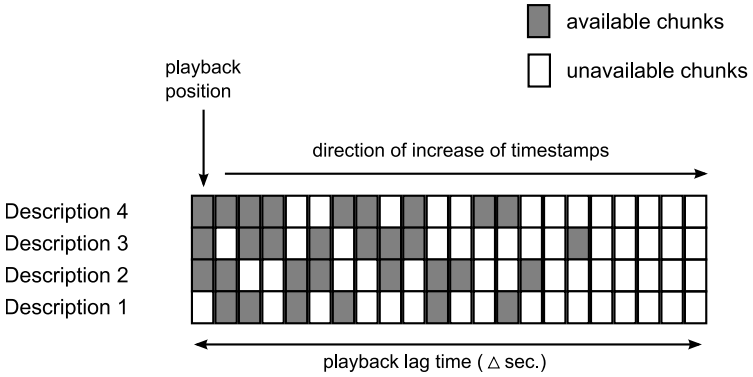


Figure 4.2: Buffer map of a peer.

ensures that it uploads every chunk the same number of times, which increases the likelihood that all chunks are delivered to peers before their playback time. A peer updates its buffer map at its neighbour by sending control messages to the neighbour.

Parallel downloads. A peer selects different chunks, requests them and subsequently downloads them from its senders in parallel. In turn, the transmitter selects chunks that may not be different from each other and uploads them to its receivers in parallel. Thus, each peer may download different chunks from multiple senders in parallel, and each peer and the transmitter may upload chunks to multiple receivers in parallel. Such parallel downloads/uploads have many advantages. In particular, downloading chunks from multiple senders in parallel allows for consistently high aggregate download rate in the light of failure of senders and network fluctuations. Moreover, uploading chunks to multiple receivers in parallel enables better utilisation of the uplink bandwidth.

Algorithms for scheduling transmission of chunks. When a media stream is encoded at a single rate, a peer must download all chunks to deliver continuous playback. However, a peer may not be able to download all chunks when its aggregate download rate from senders is below the single stream rate. By using MDC, MeshTV enables a

peer to download a subset of chunks and deliver continuous playback at reduced quality. MeshTV proposes algorithms for a peer to select this subset of chunks so that the quality of playback adapts to its download rate. Additionally, these algorithms allow for a short playback startup delay at peers joining the system.

When multiple chunks needed by a peer are available at its sender, the peer needs to decide which chunk to download first. In the client-server approach to streaming, a missing chunk with the closest playback time is generally downloaded in the first place to ensure that chunks are delivered before their playback time. However, P2P approaches require that peers cooperate with each other for the benefit of the whole system, rather than for their individual benefit. MeshTV proposes an algorithm for a peer to select the order of chunk downloads and for the transmitter to select the order of chunk uploads so that the dissemination times of chunks in the overlay are low. Low dissemination times of chunks increase the likelihood that chunks are delivered to peers before their playback time.

Algorithms for adapting the mesh overlay. In Section 3.4.2, we showed that when a random mesh overlay is used for live streaming, the uplink bandwidth of peers is underutilised and download rates vary among peers. Consequently, the quality of playback is reduced and varies among viewers. MeshTV proposes novel algorithms for peers to adapt the mesh overlay so that download rates are nearly uniform among peers and the upload rate of peers is maximised.

4.2 Chunk Distribution

The transmitter produces a media stream and splits it into chunks. Participating peers and the transmitter cooperate to distribute these chunks in the overlay.

4.2.1 Parallel Downloads

A peer in MeshTV requests and downloads different data chunks from its senders in parallel. Whenever a sender finishes transmitting a data chunk, the peer issues a new request for a chunk that it has not yet requested from another sender. As long as a sender has chunks needed by the peer, chunks are continuously transmitted with the exception of the time interval between the time when the sender completes transmission of a chunk to the peer and the time when the sender receives a subsequent request from the same peer. To eliminate this idle time, MeshTV uses pipelining [102] by which a peer may issue multiple requests for different chunks to a single sender. Thus, when the sender completes the transfer of a chunk, it may immediately initiate the transfer of a subsequent pipelined chunk. To ensure that requests are distributed across all senders, a peer limits the number of requests pipelined to each sender.

There are several advantages of parallel downloads with pipelining. First, delivery of chunks to a peer adapts to variations in the available bandwidth of the connections to its senders. In particular, the load of delivering chunks to the peer is shared by senders in a way that is proportional to the bandwidth of the connection to each sender, therefore performing automatic load balancing. Faster senders deliver larger number of chunks, while slower senders deliver smaller number of chunks. This load balancing is performed without information about the available bandwidth to senders, which is in contrast to CoolStreaming and PRIME that require such information. Second, downloading from multiple senders in parallel is more resilient to failures of senders and congestion than downloading from a single sender. Load is automatically shifted from congested parts of the Internet to parts with more available resources. And finally, the uplink bandwidth of a peer may be better utilised as multiple receivers download from the peer in parallel, performing load balancing.

4.2.2 Control Messages

To allow for distribution of chunks in the mesh overlay, peers and the transmitter send the following control messages to their neighbours:

- **BUFFER MAP.** When a peer becomes a sender of another peer, it sends the BUFFER MAP message to provide the new receiver with its buffer map, and thereby to enable the new receiver to request chunks. In turn, when the transmitter becomes a sender of a peer, the BUFFER MAP message is sent by the peer to the transmitter rather than in the opposite direction. The message is sent to provide the transmitter with the buffer map of the new receiver, and thereby to enable the transmitter to upload chunks needed by the new receiver. The BUFFER MAP message contains a bitmap consisting of 0's and 1's representing the current buffer map of the peer sending the message.

- **NOTIFY.** Whenever a peer downloads a new chunk, it sends the NOTIFY message to each receiver so that receivers can request this new chunk. This NOTIFY message contains the description number and the timestamp of the downloaded chunk. When a peer receives this message, it updates the local buffer map of the relevant sender. The NOTIFY message may also have another purpose. If the transmitter is one of the senders of the peer, the peer sends the NOTIFY message to the transmitter whenever it requests a chunk from another sender. The reason for sending this message is to prevent the transmitter from uploading the same chunk to the peer. This is further discussed in the next section.

- **REQUEST.** A peer sends the REQUEST message to request a chunk from a sender, unless the sender is the transmitter. If the transmitter is one of the senders of a peer, the peer does not request chunks from the transmitter, but instead, the transmitter selects chunks and uploads them to the peer. The REQUEST

message contains the description number and the timestamp of the requested chunk.

- **CHUNK.** The CHUNK message is used to transmit a chunk between neighbours. A peer sends this message to a receiver upon receiving the REQUEST message from the receiver. Moreover, the transmitter sends the CHUNK message to upload a selected chunk to its receiver. The message contains the description number, the timestamp and media content of the chunk being transmitted.

4.2.3 Scheduling Transmission of Chunks

Scheduling transmission of chunks refers to algorithms for peers to select chunks to download from their senders and for the transmitter to select chunks to upload to its receivers.

Adapting the Quality of Media to Download

MeshTV uses MDC to allow a peer to deliver continuous playback despite the peer not downloading all chunks. The transmitter produces M descriptions and splits each description into consecutive chunks. To decode the media stream at timestamp t at quality level m, where $m = 1, ..., M$, a peer must download at least m distinct chunks with timestamp t. To deliver playback at the quality level that is stable, meaning that this level does not oscillate significantly over short periods of time, each peer maintains the $TargetQuality$ value. The $TargetQuality$ value determines the number of distinct chunks that the peer strives to download for each timestamp and ahead of the playback time. In fact, a peer may download chunks in $TargetQuality + 1$ descriptions per timestamp when it has downloaded all chunks that are available at senders and required for $TargetQuality$ chunks per timestamp. These additional chunks are used to "probe" whether the download rate is sufficient to increase $TargetQuality$, and

consequently, sufficient to download more chunks.

Adapting TargetQuality

Algorithm 1 illustrates how each peer adapts $TargetQuality$ to its download rate. Initially, when a peer joins the system, $TargetQuality$ is set to 1. Whenever a peer receives a new chunk, it determines whether to increase $TargetQuality$. $TargetQuality$ is increased if progress of the peer at the quality level $TargetQuality + 1$ is longer than a certain threshold γ, where γ is below the playback lag time Δ. Progress at the quality level m is defined as the maximum duration (in seconds) of the media stream which starts at the current playback time and which can be decoded at the quality level m using already downloaded chunks. Function 2 illustrates how a peer calculates its progress. The function first calculates the maximum number of consecutive timestamps, starting from the current playback timestamp, such that for each of these timestamps the peer has downloaded at least m distinct chunks. This number of timestamps is then converted to the duration of the corresponding stream. This step takes into account that one timestamp corresponds to the stream duration of $ChunkSize * M/StreamRate$, where $ChunkSize$ is the size of each chunk, M is the total number of descriptions and $StreamRate$ is the sum of the rate of all descriptions.

Each peer periodically, every several seconds, checks whether its average download rate is sufficiently high to download all chunks required for the $TargetQuality$ quality level. The peer calculates the maximum quality level that can be supported at the current download rate. This maximum quality level is calculated as $\lfloor DownloadRate * M/StreamRate \rfloor$, where $DownloadRate$ is the average download rate since the last calculation. If the current $TargetQuality$ is higher than this maximum, it is reduced to the calculated maximum.

The download rate of a peer is calculated as an average over a certain time interval because it fluctuates as a result of network congestion. The length of this time

Algorithm 1 Adaptation of $TargetQuality$.

Whenever a new chunk is downloaded:
if $progress(TargetQuality + 1) > \gamma$ **then**
  $TargetQuality \leftarrow TargetQuality + 1$
end if

Execute periodically:
$max \leftarrow \lfloor DownloadRate * M/StreamRate \rfloor$
if $TargetQuality > max$ **then**
  $TargetQuality \leftarrow max$
end if
if $TargetQuality < 1$ **then**
  $TargetQuality \leftarrow 1$
end if

Function 2 $progress(m)$: Calculates progress at the quality level m.

$t \leftarrow PlaybackTimestamp$
loop
  if (number of buffered chunks with timestamp t) $< m$ **then**
    return $ChunkSize * (t - PlaybackTimestamp) * M/StreamRate$
  end if
  $t \leftarrow t + 1$
end loop

interval needs to meet two opposing goals. First, it should be long to avoid reducing $TargetQuality$ by a peer when the download rate decreases only for a short time. Second, it should be short to quickly reduce $TargetQuality$ when the download rate decreases for a long time. If the download rate decreases for a long time and $TargetQuality$ is not reduced, playback may be interrupted. As it is unknown in advance whether a download rate decrease will be temporary or lasting, the time interval for checking the download rate is uniform in MeshTV. When the download rate decreases for a time longer than this uniform time interval, the algorithm reduces $TargetQuality$ of the peer. The top part of Algorithm 1, which is executed whenever a new chunk is downloaded, increases $TargetQuality$ when the download rate increases.

An alternative approach could use variable time intervals for checking the download rate. However, when the time interval grows, a sudden decrease in the download rate may not be followed by a timely reduction of *TargetQuality* and thereby may cause interruptions of playback.

In addition to adapting the number of descriptions to download, the described algorithms allow for a short playback startup delay. This is because a peer initially downloads a single chunk per each timestamp as *TargetQuality* is initially set to 1. This amount of chunks can be downloaded quickly, offering continuous playback at a basic quality. Once the peer delivers playback of the basic quality, it gradually increases the number of chunks to download per each timestamp.

Selecting the Order of Chunk Transmissions

We propose two algorithms: one for a peer to select a chunk to request from its sender that is also a peer and one for the transmitter to select a chunk to upload to its receiver. These algorithms aim at replicating the rarest chunks in the overlay as these chunks are the most likely to be needed by peers. In P2P live streaming, the rarest chunks are typically the most recent chunks as they had the least time to be disseminated in the overlay. Therefore, in MeshTV, peers and the transmitter preferentially download and upload chunks that are *the most recent*, i.e., chunks with the highest timestamp.

Peer. Algorithm 3 is executed by a peer to select a chunk to request from its sender that is also a peer rather than the transmitter. The algorithm is executed whenever the number of requests pipelined to a sender is below the limit. In the algorithm, a peer selects the chunk with the highest timestamp (i.e., the most recent chunk) among all suitable chunks available at the sender. A chunk is suitable for a peer if it has not been downloaded or requested by the peer and if the peer downloaded or requested less than *TargetQuality* chunks with the same timestamp as the timestamp

Algorithm 3 Peer: Selection of a chunk to request from a sender.

```
for m ← TargetQuality to TargetQuality + 1 do
  t ← timestamp of the most recent chunk available at the sender
  while t > PlaybackTimestamp do
    if (number of buffered or requested chunks with timestamp t) < m then
      for desc ← 1 to M do
        if chunk < desc, t > is neither buffered nor requested and is available at
        the sender then
          return < desc, t >
        end if
      end for
    end if
    t ← t − 1
  end while
  if progress(m) ≤ γ then
    return
  end if
end for
```

of the chunk. Such a chunk may not be found if the peer has already downloaded or requested all chunks available at the sender in up to $TargetQuality$ descriptions per timestamp. In this case and when the progress of the peer at $TargetQuality$ quality level is above a certain threshold γ, the peer repeats the same selection procedure, but this time it may select a chunk with the timestamp for which it already has exactly $TargetQuality$ distinct chunks. This second iteration is used as an exploratory action. Rather than not downloading any chunks from the sender, the peer may download chunks in $TargetQuality + 1$ descriptions per timestamp. In turn, this may result in Algorithm 1 increasing $TargetQuality$ of the peer.

Transmitter. During our initial experiments, we observed that when receivers of the transmitter used Algorithm 3 to select chunks to request from the transmitter, some chunks were never uploaded by the transmitter due to the randomness of this selection. As a consequence, these chunks were not delivered to any peer in the overlay. Moreover,

Algorithm 4 Transmitter: Selection of a chunk to upload to a receiver.

```
t ← timestamp of the most recent chunk
min ← ∞
while t > PlaybackTimestamp do
  for desc ← 1 to M do
    if (number of times chunk < desc, t > has been uploaded) < min then
      if receiver has not downloaded or requested chunk < desc, t > then
        if receiver downloaded or requested less than its TargetQuality chunks
        with timestamp t then
          min ← numer of times chunk < desc, t > has been uploaded
          desc' ← desc
          t' ← t
        end if
      end if
    end if
  end for
end while
return < desc', t' >
```

some chunks were uploaded by the transmitter less often than other chunks, resulting in uneven dissemination of chunks in the overlay. Less often uploaded chunks had longer dissemination times compared to other chunks. Longer dissemination times of these chunks may prevent their delivery to peers before their playback time. To address this, in MeshTV, chunks are not requested from the transmitter, but instead, the transmitter selects chunks and uploads them to its receivers. By doing so, the transmitter ensures that it uploads each chunk approximately the same number of times. This significantly increases the likelihood that all chunks are evenly disseminated in the overlay.

Algorithm 4 is executed by the transmitter to select a chunk to upload to a receiver whenever the transmitter completes a previous upload to the same receiver. The chunk to upload is selected so that it has the highest timestamp among those chunks suitable for the receiver that the transmitter uploaded the minimum number of times. As in the previous algorithm, a chunk is suitable for a receiver if the chunk has not been downloaded or requested by the receiver from other senders and if the receiver

downloaded or requested less than receiver's *TargetQuality* of other chunks with the same timestamp.

The chunk selection algorithm for the transmitter requires that the transmitter knows about chunks already downloaded or requested by its receivers and also about their current *TargetQuality* value. To achieve this, the transmitter maintains the buffer map of each receiver, whereas receivers notify it whenever they request a chunk from their other senders. The receivers do it by sending the NOTIFY message, similar to how they notify their receivers when they download a new chunk. Each NOTIFY message sent to the transmitter contains the receiver's current *TargetQuality* value in addition to the description number and the timestamp.

In the algorithm, the transmitter selects and uploads each chunk approximately the same number of times. This significantly increases the likelihood that chunks are disseminated evenly in the overlay and that their dissemination times are nearly uniform. However, receivers of the transmitter may receive duplicated chunks. This may happen if the transmitter uploads a chunk to a receiver at roughly the same time when the receiver requests the same chunk from another sender. However, this is unlikely for two reasons. First, the time interval when this situation may occur is short as it corresponds to the packet transmission time between the transmitter and the receiver. Second, the transmitter typically uploads chunks that other senders do not possess. The transmitter uploads the least uploaded chunks, which are typically the most recent chunks, and thus the rarest among peers.

4.3 Mesh Overlay Adaptation

In this section, we present algorithms for adapting the mesh overlay so that download rates are nearly uniform among peers and maximise the usage of the uplink bandwidth of peers. First, we outline objectives with respect to the download rate of peers as well

as objectives with respect to the operation of the algorithms. Then, we show properties of a mesh overlay that allow download rates to meet the objectives. Subsequently, we present our basic algorithm for adapting a mesh overlay so that the overlay exhibits these properties. This algorithm is then enhanced to improve its performance. And finally, we describe the membership management protocol used in MeshTV.

4.3.1 Objectives

Objectives of the MeshTV overlay adaptation with respect to peer download rates are:

- Utilise the maximum of the available uplink bandwidth of all peers and the transmitter for the dissemination of content. Formally,

$$\sum_{i=1}^{N} DownloadRate_i = \min\left(\sum_{i=0}^{N} uplink_i, \sum_{i=1}^{N} downlink_i\right) \tag{4.1}$$

  where $DownloadRate_i$, $uplink_i$ and $downlink_i$ are, respectively, the download rate, the uplink bandwidth and the downlink bandwidth of peer i. Indices range from 0 to N, where index 0 is reserved for the transmitter. The transmitter contributes its uplink bandwidth, but does not download content, so its download rate is equal to 0. This formula captures the possibility that the downlink bandwidth of some peers may reduce their download rate. If the cumulative downlink bandwidth of peers is lower than the cumulative uplink bandwidth of all peers and the transmitter, then peers cannot utilise the entire uplink bandwidth available in the system. However, this is unlikely due to the popularity of asymmetric Internet connections, such as ADSL, where the downlink bandwidth is higher than the uplink bandwidth.

- Deliver content at the same rate to each peer, unless the download rate of the

peer is reduced by the peer's downlink bandwidth. Formally,

$$DownloadRate_i = \min\left(downlink_i, \max_{1 \leq j \leq N} DownloadRate_j\right) \text{ for all } i = 1, ..., N \tag{4.2}$$

This formula also captures the possibility that the individual downlink bandwidth of a peer may reduce its download rate. If this is the case, the portion of the uplink bandwidth unused by the peer is shared among remaining peers in order to increase their download rate.

When these two objectives are satisfied, download rates are nearly uniform among peers and the upload rate of peers and of the transmitter is maximised. Furthermore, Formulas 4.1 and 4.2 can be simplified under the assumption that the downlink bandwidth of peers does not reduce the download rate of these peers. Then, download rates are uniform among peers and the two formulas are equivalent to:

$$DownloadRate_i = \frac{\sum_{j=0}^{N} uplink_j}{N} \text{ for all } i = 1, ..., N \tag{4.3}$$

Objectives with respect to the operation of the algorithms are:

- **Decentralisation for scalability.** A centralised adaptation algorithm requires continuous global knowledge about all peers in the system and about their currently available bandwidth. This does not scale to a large number of peers with dynamic bandwidth. To avoid scalability problems, the algorithm should be decentralised, meaning that individual peers adapt the overlay.

- **Continuous adaptation.** Arrival and departure of peers and variation in their available bandwidth may occur at any time, so the algorithm should never cease to adapt the overlay.

- **Low communication overhead.** Communication overhead incurred by the algorithm reduces the bandwidth available for the dissemination of media content.

Thus, the communication overhead should be minimised to allow for the maximum download rate of peers and thereby for the maximum quality of their playback.

- **Short adaptation time.** To maximise the quality of playback at peers in the minimal amount of time, the time required by the algorithm to adapt the overlay should be minimised.

For the purpose of algorithms presented in this section, we assume that any two peers can communicate with each other, irrespective of firewalls and Network Address Translations (NATs). Techniques discussed in [35] can be used to enable communication through most firewalls and NATs.

4.3.2 Desired Properties of the Overlay

The overlay adaptation algorithms are based on our observation that download rates may be uniform among peers and the upload rate of peers may be maximised when the overlay exhibits the following properties:

1. The number of receivers (out-degree) of each peer and of the transmitter is proportional to the uplink bandwidth. Formally,

$$\frac{uplink_i}{outdegree_i} = \frac{uplink_j}{outdegree_j} \text{ for all } i, j = 0, 1, .., N \tag{4.4}$$

   In other words, the *uplink-to-outdegree ratio* is the same at all peers and the transmitter.

2. The number of senders (in-degree) is the same at every peer. Formally,

$$indegree_i = indegree_j = K \text{ for all } i, j = 1, ..., N \tag{4.5}$$

   where K is a system-wide constant denoting the number of senders of each peer. The transmitter has no senders.

For the observation to be valid, the uplink bandwidth of a peer and of the transmitter should be shared approximately equally among receivers. To achieve this, a protocol for transmitting data over the physical network should allocate the available uplink bandwidth of a peer approximately equally among connections to receivers. This prevents the use of traditional TCP/IP congestion control protocols, such as TCP Reno, which show a bias against connections with long Round-Trip Time (RTT) [67]. Congestion control protocols that share the available bandwidth approximately equally among connections and that can be used in MeshTV have been proposed in [77, 16, 46].

Based on Formulas 4.4 and 4.5, we can derive an equation for the desired out-degree of peer i:

$$outdegree_i = \frac{uplink_i * N * K}{\sum_{j=0}^{N} uplink_j} \tag{4.6}$$

where $i = 0, 1, ..., N$. This is based on the observation that the amount of uplink bandwidth allocated by a peer to each receiver is equal to $1/K$th of the download rate of the receiver as the receiver downloads content from K senders in parallel. The download rate of the receiver is obtained from Formula 4.3. When the out-degree of a peer is smaller or larger than the right-hand side of the above equation, we say that the peer is, respectively, underloaded or overloaded.

4.3.3 Basic Overlay Adaptation Algorithm

In the basic adaptation algorithm, each peer continuously ensures that it has K senders, where K is a system-wide configurable constant. When the number of senders of a peer decreases due to departure of a sender, the peer selects a random peer as a new sender. Furthermore, each peer periodically performs the exploration action. In the exploration action, a peer replaces the sender from which it receives the worst download rate with a new, exploratory, sender selected randomly from the set of all participating peers and

the transmitter. A random peer/transmitter is provided to a peer by the membership management protocol.

There are two reasons why we chose the random peer selection approach for finding new senders of peers. First, random selection minimises the time required to find a new sender to replace an existing one that leaves the overlay. This improves resilience to peer churn compared to approaches that require finding a specific sender. Second, random selection minimises the communication overhead of finding a sender.

The exploration action requires that a peer maintains an estimated download rate, denoted as $drate_s$, for each sender s in order to determine the worst sender. For this reason, each peer measures the download rate received from each of its senders between consecutive exploration actions and updates the current estimate using the formula

$$drate_s = \alpha \times newMeasurement_s + (1 - \alpha) \times drate_s \tag{4.7}$$

where α is a weighting factor that determines the significance of the latest measurement over historical ones. The weighted update prevents disconnecting a generally well performing sender due to interim fluctuations in its performance. Such fluctuations may occur when the sender becomes temporarily overloaded by too many new receivers that selected it as an exploratory sender. Its old receivers take into account the history of its performance and may decide to stay connected to the sender despite the poor interim download rate. However, the new receivers replace the sender due to the unsatisfying download rate, thereby restoring its good performance for the old receivers.

In the algorithm, in each exploration round, each peer and the transmitter gains, on average, one new receiver. This is because the membership management protocol provides each peer with a random peer selected uniformly from the set of all participating peers and the transmitter. As a consequence, in each exploration round, each peer and the transmitter is selected as an exploratory sender, on average, by one peer.

In each exploration round, each peer, and the transmitter, may also lose some exis-

ting receivers. Each receiver of a peer, or of the transmitter, compares its download rate from this peer/transmitter with its download rate from each of the remaining $K-1$ senders. If the download rate from the peer/transmitter is the lowest, the receiver disconnects from the peer/transmitter, thereby reducing the load of this peer/transmitter.

The out-degree of each peer, and the transmitter, approaches the desired out-degree determined by Equation 4.6. When a peer, or the transmitter, is underloaded, it gains, on average, one new receiver in each exploration round and its existing receivers do not disconnect, so its out-degree increases. When a peer, or the transmitter, is overloaded, it gains, on average, one new receiver, but some existing receivers may disconnect, so its out-degree stays constant or decreases. Typically, the out-degree of a peer increases until it reaches the desired out-degree and then it oscillates around this desired out-degree.

A drawback of the presented overlay adaptation algorithm is that the out-degree may increase, on average, by only 1 in each exploration round. This may result in a long time of adaptation of the out-degree of peers with high uplink bandwidth. For example, for $K = 10$ and the heterogeneous bandwidth distribution used in our experiments (see Table 5.1), the desired out-degree of peers with the highest uplink bandwidth is equal to 42. When such a peer arrives in the system, it initially has no receivers, and so it needs, on average, at least 42 exploration rounds to gain the desired number of receivers.

4.3.4 MeshTV Overlay Adaptation Algorithm

In this section, we enhance the basic algorithm to improve the speed of overlay adaptation. In the enhanced algorithm, a peer and the transmitter may gain multiple receivers in each exploration round. To achieve this, we modify how a peer discovers an exploratory sender and how a peer selects an existing sender to disconnect.

Similarly to the basic algorithm, each peer continuously ensures that it has K

senders. A peer also periodically replaces one of its existing senders with an exploratory sender. We first describe how a peer discovers an exploratory sender and then how it selects an existing sender to disconnect.

Discovery of an Exploratory Sender

The discovery of an exploratory sender for peer A involves the following steps:

1. Peer A selects a random peer R.
2. Peer A requests peer R for one of its senders.
3. Peer R selects one of its senders and returns the selected sender to peer A.
4. The returned sender of R becomes the exploratory sender of peer A.

To find a random peer in step 1, peer A uses the membership management protocol. The membership management protocol provides a peer selected at random or the transmitter selected with the same probability as any peer. The transmitter has no senders, so if it is provided by the membership management protocol, it becomes the exploratory sender of the requesting peer and the remaining steps 2-4 are skipped.

In step 3, the random peer R selects one of its senders to be returned to peer A. By selecting a sender, peer R increases the out-degree of the sender (peer A becomes a new receiver of the sender) and thereby reduces the uplink-to-outdegree ratio of the sender. As the goal of the overlay adaptation is to equalise the uplink-to-outdegree ratio of all peers, peer R selects the sender so that the uplink-to-outdegree ratio of its senders is maximally close to each other.

It is not always appropriate for peer R to select the sender with the highest uplink-to-outdegree ratio. A selection strategy based on selecting always the sender with the highest uplink-to-outdegree ratio results in large oscillations in the uplink-to-outdegree ratio of peers over time. This is because a peer with the uplink-to-outdegree ratio

insignificantly higher than the uplink-to-outdegree ratio of other peers might be selected by all receivers. Consequently, the number of receivers of such a peer may double. To avoid such oscillations, in MeshTV, peer R selects a sender probabilistically, where the probability of selecting each sender corresponds to the uplink-to-outdegree ratio of the sender.

The strategy for peer R to select a sender is based two observations. First, the random peer may estimate the uplink-to-outdegree ratio of each sender based on the download rate from the sender as these two values are approximately the same. Second, if each receiver of a peer selects this peer to be returned as an exploratory sender with the same probability p, then the out-degree of the peer, equal to $outdegree$, is expected to change to

$$outdegree * (1 + p) \tag{4.8}$$

Correspondingly, the download rate from this peer at each receiver, equal to $drate$, is expected to change to

$$\frac{drate}{1 + p} \tag{4.9}$$

In MeshTV, a peer individually computes probability p_i of selecting each sender i, for $i = 1, 2, ..., K$. Given the download rate from each sender, $drate_i$, the peer computes each p_i such that $0 \leq p_i \leq 1$, $\sum_{i=1}^{K} p_i = 1$ and the expected new download rates, $drate_i/(1 + p_i)$, are maximally close to each other. Notice that by increasing p_i, the value of $drate_i/(1 + p_i)$ decreases, and so the values of p_i should minimise

$$\max_{1 \leq i \leq K} \frac{drate_i}{1 + p_i} \tag{4.10}$$

To derive the algorithm for finding each p_i, we define the set MAX that contains all indices i for which $drate_i/(1 + p_i)$ is equal to the value of Formula 4.10. Notice that to minimise the value of Formula 4.10, p_i may be greater than 0 only for $i \in MAX$. If p_j was greater than 0 for some $j \notin MAX$, then by sharing the value of p_j among

all p_i for $i \in MAX$, the value of Formula 4.10 could be further reduced. Thus, the set MAX needs to satisfy the following conditions:

$$\frac{drate_i}{1+p_i} > drate_j \text{ for all } i \in MAX, j \notin MAX \tag{4.11}$$

$$\frac{drate_i}{1+p_i} = \frac{drate_j}{1+p_j} \text{ for all } i, j \in MAX \tag{4.12}$$

for p_i ($i = 1, .., K$ and $0 \leq p_i \leq 1$) such that

$$\sum_{i \in MAX} p_i = 1 \tag{4.13}$$

To calculate p_i, we substitute p'_i for $1 + p_i$ into Equations 4.12 and 4.13:

$$\frac{drate_i}{p'_i} = \frac{drate_j}{p'_j} \text{ for all } i, j \in MAX \tag{4.14}$$

$$\sum_{i \in MAX} p'_i = |MAX| + 1 \tag{4.15}$$

where $|MAX|$ is the number of elements of the set MAX. From these, we derive

$$p'_i = (|MAX| + 1) * \frac{drate_i}{\sum_{j \in MAX} drate_j} \text{ for all } i \in MAX \tag{4.16}$$

And if we substitute back $1 + p_i$ for p'_i, we obtain

$$p_i = (|MAX| + 1) * \frac{drate_i}{\sum_{j \in MAX} drate_j} - 1 \text{ for all } i \in MAX \tag{4.17}$$

This equation for p_i can be substituted into the condition in Formula 4.11

$$\frac{\sum_{i \in MAX} drate_i}{|MAX| + 1} > drate_j \text{ for all } j \notin MAX \tag{4.18}$$

Algorithm 5 uses these formulas to compute each p_i, given $drate_i$, for $i = 1, 2, ..., K$. In lines 1-4, the algorithm finds the set MAX by iteratively adding indices $i \notin MAX$ for which $drate_i$ is maximal. This continues until MAX contains all indices or the condition in Formula 4.18 is satisfied. In lines 5-7, p_i for all $i \in MAX$ are set according

Algorithm 5 Computes p_i, given $drate_i$, for all $i = 1, 2, ..., K$ such that $0 \leq p_i \leq 1$, $\sum_{i=1}^{K} p_i = 1$ and $\max_{1 \leq i \leq K}(drate_i/(1 + p_i))$ is minimal.

1: $MAX \leftarrow \emptyset$
2: **repeat**
3: $\quad MAX \leftarrow MAX \cup \arg\max_{i \notin MAX} drate_i$
4: **until** $|MAX| = K$ or $\max_{i \notin MAX} drate_i < \frac{\sum_{i \in MAX} drate_i}{|MAX| + 1}$
5: **for** $i \in MAX$ **do**
6: $\quad p_i \leftarrow (1 + |MAX|) * \frac{drate_i}{\sum_{j \in MAX} drate_j} - 1$
7: **end for**
8: **for** $i \notin MAX$ **do**
9: $\quad p_i \leftarrow 0$
10: **end for**

to Equation 4.17. In lines 8-10, remaining p_i are set to 0. When the algorithm finishes, $\max_{1 \leq i \leq K}(drate_i/(1+p_i))$ is minimal and probabilities p_i may be used to select a sender to be returned to the requesting peer.

The described method of the discovery of an exploratory sender does not specify how peers acquire their very first receiver. If a peer has no receivers, it cannot be discovered as a sender of a random peer. Therefore, to acquire an initial receiver, in step 3, the random peer R with no receivers returns to the requesting peer A itself rather than one of its senders. As a consequence, the requesting peer A becomes the first receiver of the random peer R and may subsequently provide more receivers.

Selection of a Sender to Disconnect

To keep a constant number of senders, a peer disconnects one of its existing senders before connecting to an exploratory sender obtained from a random peer. The sender to disconnect is selected with the objective to equalise the uplink-to-outdegree ratio of senders. Disconnection of a sender decreases its out-degree, and so increases its uplink-to-outdegree ratio. To prevent large oscillations in the out-degree of peers over time,

Algorithm 6 Computes p_i, given $drate_i$, for all $i = 1, 2, ..., K$ such that $0 \leq p_i \leq 1$, $\sum_{i=1}^{K} p_i = 1$ and $\min_{1 \leq i \leq K}(drate_i/(1 - p_i))$ is maximal.

1: $MIN \leftarrow \emptyset$
2: **repeat**
3: $\quad MIN \leftarrow MIN \cup \arg\min_{i \notin MIN} drate_i$
4: **until** $|MIN| = K$ or $\min_{i \notin MIN} drate_i > \frac{\sum_{i \in MIN} drate_i}{|MIN| - 1}$
5: **for** $i \in MIN$ **do**
6: $\quad p_i \leftarrow 1 - (|MIN| - 1) * \frac{drate_i}{\sum_{j \in MIN} drate_j}$
7: **end for**
8: **for** $i \notin MIN$ **do**
9: $\quad p_i \leftarrow 0$
10: **end for**

the sender to disconnect is selected probabilistically, similar to how an exploratory sender is selected by a random peer.

The strategy for selecting the sender to disconnect is based on the observation that if each receiver of a peer disconnects this peer with the same probability p, then the out-degree of the peer, equal to $outdegree$, is expected to change to

$$outdegree * (1 - p) \tag{4.19}$$

Correspondingly, the download rate from this peer at each receiver, equal to $drate$, is expected to change to

$$\frac{drate}{1 - p} \tag{4.20}$$

A peer individually computes probability p_i of disconnecting each sender i, for $i = 1, 2, ..., K$. Given the download rate from each sender, $drate_i$, the peer computes each p_i such that $0 \leq p_i \leq 1$, $\sum_{i=1}^{K} p_i = 1$ and the expected new download rates, $drate_i/(1 - p_i)$, are maximally close to each other. Notice that by increasing p_i, the value of $drate_i/(1 - p_i)$ increases, and so the values of p_i should maximise

$$\min_{1 \leq i \leq K} \frac{drate_i}{1 - p_i} \tag{4.21}$$

Algorithm 6 shows how each p_i is computed. We do not discuss the algorithm as it is analogous to Algorithm 5.

4.3.5 Membership Management

In this section, we discuss the membership management protocol that is used in MeshTV to provide peers with addresses of a random subsets of all peers and the transmitter. Membership management may be realised as a centralised component. For example, BitTorrent uses a centralised tracker that maintains a list of all participating peers. All peers periodically notify the tracker about their presence. Peers that do not notify the tracker are removed from the list. When needed, peers may request a random subset of all peers from the tracker. However, for large P2P overlays, a centralised tracker may become a system bottleneck. For this reason, more recent versions of BitTorrent incorporate DHT overlays to decentralise the tracker service [36].

Distributed approaches to membership management have been a focus of much recent work in P2P systems. Structured and unstructured P2P overlays have been used for membership management. Approaches using structured overlays include DHT-based [17] and tree-based [62]. Approaches using unstructured overlays are typically based on gossipping [126, 58, 37, 59] or random walks [123]. Gossip-based approaches are especially suitable for applications that require that each peer periodically receives a fresh sample of peer addresses. MeshTV is an example of such application as it uses the overlay adaptation algorithm that requires that each peer periodically discovers a random peer. For such applications, gossip-based approaches offer low communication overhead and high resilience to peer churn [37].

In gossip-based approaches, each peer periodically produces a fresh *partial view* that contains addresses of a limited subset of all peers in the system. These peer addresses may be used by a peer to find initial or new neighbours. For many applications, including MeshTV, it is important that the distribution of peer addresses among partial

views is nearly uniform. If the distribution is not uniform, addresses of some peers may appear more often in partial views than addresses of other peers. This may result in some peers being selected as neighbours more often than other peers, and potentially, becoming overloaded. Many existing gossip-based protocols have been shown to produce non-uniform distributions [58].

In MeshTV, we selected the CYCLON protocol [126] for gossip-based membership management. The reason for selecting CYCLON is that it generates nearly uniform distributions of peer addresses among partial views. These distributions have even lower variance than a uniform random distribution that is equivalent to selecting peer addresses at random for inclusion in partial views.

In CYCLON, each peer maintains a partial view that contains at most c entries, where c is a system-wide configurable parameter with a typical value of 20 or 50. Each entry contains the network address of a participating peer and the age of the entry. Each peer P periodically initiates the *enhanced shuffle* algorithm, proposed in [126], that performs the following steps:

1. Increase by one the age of all entries in the partial view.
2. Select peer Q with the highest age among all entries in the partial view and select additional $l-1$ ($1 \leq l \leq c$) other random entries.
3. Replace Q's entry with the new entry that has age 0 and P's address.
4. Send the updated subset of size l to peer Q.
5. Receive from Q a subset of no more than l of Q's entries.
6. Discard entries with P's address and entries already contained in P's partial view.
7. Include the remaining received entries in the partial view. Some entries sent to Q may have to be removed from the partial view so that the number of entries in the partial view does not exceed c.

The parameter l is system-wide and configurable. The receiving peer Q replies by sending back a random subset of at most l of its own entries and updates its partial view to include all received entries. Peer Q does not increase the age of any entry until its own turn comes to initiate the shuffle algorithm.

Furthermore, CYCLON proposes a method for joining peers to obtain their initial partial views. However, this requires that a joining peer knows a single peer that is already participating in the protocol. Such a peer may be discovered in various ways, including broadcasting in the local network, making use of IP multicast, or contacting a well-known server. Finding such a peer is beyond the scope of this work and has been addressed elsewhere [41].

4.3.6 Discussion

In this section, we first presented objectives of the MeshTV overlay adaptation with respect to the download rate of peers and with respect to the operation of the algorithms. Then, we presented properties of the mesh overlay that allow download rates to meet the objectives. We presented an initial algorithm that adapts the mesh overlay so that the overlay exhibits these properties. In the algorithm, the in-degree of each peer is constant, whereas the out-degree of a peer and the transmitter adapts to the uplink bandwidth. However, we identified that in this initial algorithm, the out-degree of a peer may increase, on average, at most by 1 in each exploration round. To improve performance, we devised an enhanced algorithm in which the out-degree of a peer may increase faster. Experimental results, presented in the next chapter, show that the enhanced algorithm may adapt a random overlay in only 4 exploration rounds and that this adaptation time is independent of the number of participating peers.

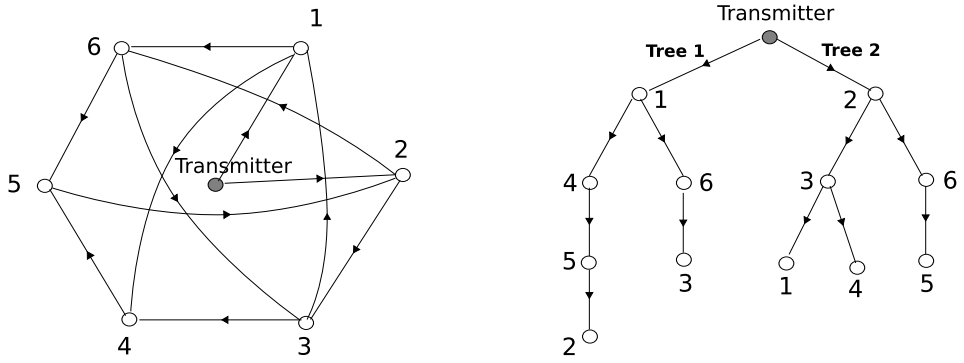
The presented overlay adaptation algorithms satisfy our further objectives with respect to their operation. The algorithms are decentralised to ensure their scalability. To accommodate dynamic arrival and departure of peers and variations in peer bandwidth,

the algorithms never cease to adapt the overlay. Moreover, the algorithms require low communication overhead. In the initial algorithm, the communication overhead of a peer is related to disconnecting a single sender and connecting to a single new sender in each exploration round. In the enhanced algorithm, the communication overhead of a peer involves additionally requesting an exploratory sender from a random peer in each exploration round.

Limited downlink bandwidth. When the downlink bandwidth of some peers reduces their download rate, a portion of the uplink bandwidth allocated to these peers by their senders is unused by these peers. This unused portion of the uplink bandwidth should be used to increase the download rate of remaining peers in the overlay. In particular, the download rate of remaining peers should be increased by an approximately equal amount so that download rates are uniform among these peers.

To show that MeshTV adapts the overlay such that the uplink bandwidth unused by peers with low downlink bandwidth increases the download rate of remaining peers, assume that peer P has a receiver R and that the downlink bandwidth of R reduces R's download rate from P. A congestion control protocol shares the available uplink bandwidth of P equally among its receivers. Since R cannot use a portion of P's uplink bandwidth that has been allocated to it, the congestion control protocol shares this unused portion of the uplink bandwidth among remaining receivers of P. This increases the download rate from P at each of the remaining receivers of P. However, when the download rate from P at these receivers is higher than their download rate from other senders, they are more likely to return P to peers requesting an exploratory sender. Thereby, the number of receivers of P may increase and, eventually, download rates become uniform among peers with sufficient downlink bandwidth.

Stream reception delay. The stream reception delay of a peer is the delay between the time when chunks of the media stream are produced by the transmitter and the



(a) Mesh overlay for K=2.

(b) Corresponding two multicast trees.

Figure 4.3: Representation of the mesh overlay as multiple multicast trees.

time when these chunks are received by the peer. In a single multicast tree, the stream reception delay of a peer is proportional to the depth of the peer in the tree. To minimise the stream reception delay of peers, the depth of the tree needs to be minimised. To accomplish this, the tree is constructed so that peers with a higher out-degree (i.e., with higher uplink bandwidth) are placed at a lower depth (i.e., closer to the transmitter).

In multiple multicast trees, the stream reception delay of a peer is proportional to its maximum depth in a tree. Therefore, to reduce the stream reception delay of peers, the depth of all trees needs to be minimised. To accomplish this, multicast trees are constructed so that each peer is an interior node in at most one tree and peers with a higher out-degree are placed at a lower depth in each tree.

In contrast to single and multiple multicast trees, mesh overlays are unstructured and thereby more difficult to analyse. However, the adapted mesh overlay may be represented as multiple multicast trees [13, 76]. In this representation, each of K senders of a peer in the mesh overlay is a parent of this peer in one of the trees, as illustrated in Figure 4.3. Thus, the stream reception delay of peers can be minimised in the same way as it is accomplished in multiple multicast trees. As a peer should be

an interior node in at most one tree and a leaf node in the remaining trees, only one sender of the peer in the mesh overlay should be close to the transmitter, whereas the remaining senders should be further away. Moreover, the distance from the transmitter to this single sender should be lower for peers with a higher out-degree. Work on an algorithm for such adaptation of the mesh overlay has been initiated in [13] and remains open for future work.

4.4 Discussion

In this chapter, we described the MeshTV system and its algorithms. In particular, we presented the algorithm for a peer to select chunks to download so that the quality of playback adapts to the cumulative download rate of the peer from its senders. This algorithm also allows for a short playback startup delay. A peer initially downloads a small number of chunks that correspond to the basic quality of playback and allow for a short startup delay. The quality of playback gradually improves over time as the number of chunks to download is increased. We also presented the algorithm for a peer to select a chunk to request from a sender and for the transmitter to select a chunk to upload to a receiver. These algorithms disseminate chunks in the overlay so that the dissemination time is uniform among chunks, increasing the likelihood that chunks are delivered to peers before their playback time.

Furthermore, we presented algorithms for peers to adapt the mesh overlay so that download rates are nearly uniform among peers and the upload rate of peers is maximised. When these algorithms are combined with algorithms for adapting the quality of playback, peers deliver playback at the quality that is nearly uniform among peers and maximises the uplink bandwidth available in the system.

Chapter 5

Evaluation

In this chapter, we evaluate the MeshTV system to show that it is scalable, resilient, delivers playback at the optimal quality to all peers, and allows for a short playback startup delay. This chapter is organised as follows. In Section 5.1, we discuss various approaches for evaluating P2P systems and reasons for the use of a simulator to evaluate MeshTV. In Section 5.2, we present the MeshTV network simulator. In Section 5.3, we discuss settings common for experiments in this chapter.

In Section 5.4, we show that MeshTV adapts the overlay so that the out-degree of each peer adapts to the peer's uplink bandwidth, the upload rate of each peer is maximised, and download rates are nearly uniform among peers. We also show that these download rates are resilient to peer churn and to catastrophic failures. We analyse the delay between the time when the stream is produced at the transmitter and the time when it is delivered to each peer. We show that this delay is below 7 seconds for 5000 peers and is nearly uniform among peers. We show that a random overlay adapts in only 4 adaptation rounds and that this adaptation time is independent of the number of participating peers, thereby ensuring the scalability of overlay adaptation. We analyse the communication overhead of MeshTV and show that it is about 13% of the media content transmitted in the network.

Furthermore, in Section 5.5, we show that the quality of playback at peers adapts to their download rate. We also show that the quality of playback is resilient to peer churn and catastrophic failures. We show that joining peers exhibit only 3 seconds delay before the start of media playback at the basic quality level. Finally, in Section 5.6, we conclude this chapter.

5.1 Approaches for Evaluating P2P Systems

The main types of approaches for evaluating large-scale P2P systems are based on real-world measurements, experimental network testbeds, analytical modelling and simulation modelling.

Real-world measurements. Real-world measurements may be used to evaluate already deployed systems that have a large population of users. Measurements typically involve collecting and analysing data about peers and about the network traffic between these peers [68, 92, 105, 50]. Measurements have the advantage that they evaluate systems in target deployment environments of these systems. However, measurements do not allow for evaluation in advance of building and deploying systems. Such early evaluation may be necessary to consider alternative system designs and algorithms.

Network testbeds. A network testbed is a computer network made available to researchers to allow them to experiment with their distributed systems [91, 130, 120]. The advantage of network testbeds is that they allow to evaluate systems on real computers and real physical networks. The disadvantages of evaluating P2P systems are following. First, widely accessible network testbeds, such as PlanetLab [91] and EmuLab [130], are used for experiments concurrently by many researchers. Consequently, network and computer resources available to each experiment are relatively small, limiting the scale of each experiment. Second, multiple experiments running in parallel

on the same computer network may interfere with each other, making their evaluation results unreproducible [113]. Reproducible results are necessary to compare different systems with each other and to consider alternative algorithms. Finally, network testbeds are not representative of the target deployment environment of P2P systems [113]. This is because network testbeds are built on high capacity managed infrastructures, whereas P2P systems are typically deployed on desktop machines connected to the Internet using DSL or cable modems.

Analytical modelling. Analytical modelling develops mathematical functions that describe how a system changes from one state to another and how variables of the system depend on each other [97, 64]. These functions allow to obtain evaluation results independently of the scale (e.g., number of peers, rate of a media stream) of the system. However, complex interactions between peers and numerous other environmental factors, such as variations in peer bandwidth or peer churn, require that analytical models often rely on simplifying assumptions that break down under real-world conditions [14].

Simulation modelling. Simulation modelling relates to evaluation of a system with a simulator. The advantage of simulation modelling compared to real-world measurements and experiments on network testbeds is that a simulator provides a controlled environment, where results are reproducible. A simulator also allows to experiment with different system and network configurations, which may not be possible using real-world measurements and network testbeds. Finally, a simulator is often easier to build compared to a real-world system that is required for experiments on network testbeds and for measurements. A disadvantage of simulation modelling is that the accuracy of evaluation results depends on the accuracy of the simulator.

A simulator of a P2P content delivery system simulates both the operation of the system and the transmission of data between peers over the underlying physical net-

work. Based on how it simulates the transmission of data over the network, it is classified as *flow-level* or *packet-level*.

Flow-level simulators. A flow-level simulator simulates the transmission of batch data over network links [43, 132, 10]. It determines the bandwidth available to a connection between two peers and transmits data in a single batch over the connection. By simulating the transmission of batch data rather than individual data packets, flow-level simulators reduce computations and memory usage. Flow-level simulators typically allocate bandwidth to connections using the method called minimum-share allocation [43]. In this method, the bandwidth allocated to a connection is given by the minimum of fair shares of bandwidth at the uplink of a sender peer and the downlink of a receiver peer. The fair share of bandwidth of a link is defined as the link bandwidth divided by the number of connections concurrently sharing the link. The advantage of this method is high computational efficiency. However, this method inaccurately allocates bandwidth to connections concurrently sharing a single network link. In particular, it may not find a possible allocation of the link bandwidth among these connections such that the entire link bandwidth is utilised. Consequently, the bandwidth available in the network may be underutilised.

The inefficiency of bandwidth allocation in flow-level simulators is illustrated in Figure 5.1. In this figure, peers X and Y upload data to peers A, B and C. The uplink bandwidth of peers X and Y is 3 Mbps each. The downlink bandwidth of peers A, B and C is respectively 3 Mbps, 0.5 Mbps and 1 Mbps. The minimum-share allocation in this scenario is following. The connection $X \rightarrow A$ is allocated 1 Mbps, which is the uplink bandwidth of peer X shared equally among its three connections with peers A, B and C. The connection $X \rightarrow B$ is allocated 0.5 Mbps, which is B's downlink bandwidth. The connections $X \rightarrow C$ and $Y \rightarrow C$ are allocated 0.5 Mbps each, which is the downlink bandwidth of peer C shared equally between its two connections with

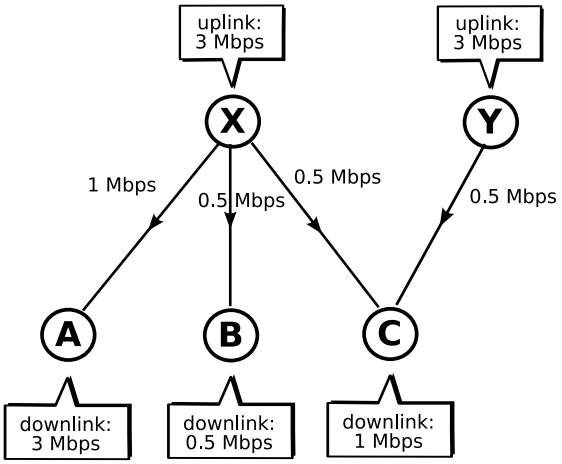


Figure 5.1: Example of inefficient bandwidth allocation to connections in a flow-level simulator.

peers X and Y. However, this bandwidth allocation results in the uplink bandwidth of peer X being underutilised. Peers B and C underuse their fair share of X's uplink bandwidth due to their constrained downlink bandwidth. This unused portion of the uplink bandwidth of peer X should be allocated to peer A, which has enough downlink bandwidth to use it. However, solving this issue would incur significant complexity in the design of flow-level simulators and would reduce their performance.

Packet-level simulators. Packet-level network simulators, such as ns-2 [84], OMNeT++ [87] and SSFNet [114], provide a more realistic network model compared to flow-level simulators. They simulate the transmission of each individual data and control packet, imposing propagation delays and bandwidth restrictions at each link. Packet-level simulators typically provide implementations of main Internet protocols, including multiple variants of the TCP/IP protocol [115]. They simulate the transmission of every data packet and control packet, such as TCP SYN, ACK. They also simulate the operation of protocol specific mechanisms, such as the TCP congestion control and TCP timeouts. However, such detailed simulations involve intensive com-

putations and large memory usage, normally limiting simulations to a maximum of hundreds of peers.

5.2 MeshTV Network Simulator

Flow-level network simulators scale to a larger number of peers, but they reduce the accuracy of bandwidth modelling compared to packet-level network simulators. As discussed, flow-level simulators may underutilise the bandwidth available in the overlay. This renders them inappropriate for evaluating the effectiveness of bandwidth utilisation in MeshTV. On the other hand, existing packet-level network simulators aim at modelling every single detail of the physical network, including routers, links and protocols. Such detailed simulations incur a large overhead and thereby limit the scalability of simulations. For example, in [12], we proposed and evaluated the basic MeshTV overlay adaptation algorithm described in Section 4.3.3 of this thesis. For the evaluation, we used the popular ns-2 simulator with the entire TCP/IP stack. However, we were unable to simulate more than 500 peers with ns-2 due to the large memory and processor consumption during these simulations. Such low number of peers is not representative for real-world live events.

To experiment with more than 500 peers, we developed our own packet-level network simulator. For the accuracy of network modelling, it simulates the transmission of every data packet over the network and imposes bandwidth constraints and propagation delay on network links. For scalability, it allocates the bandwidth of network links among connections concurrently sharing these links without the overhead of simulating congestion avoidance and control [56]. We experimentally validated the accuracy of our network simulator by comparing MeshTV evaluation results obtained using this simulator and using ns-2. The evaluation results were the same using both simulators when simulating up to 500 peers, which was the maximum number supported by ns-2.

Furthermore, we plan to release our network simulator to the public to let researchers simulate their own P2P content delivery systems at scales larger than those achievable with ns-2. We expect this will provide more feedback on the accuracy of our network simulator.

Our network simulator assumes a star topology of the physical network as illustrated in Figure 5.2. In this topology, every peer is located at an edge and is connected with its downlink and uplink to a single router in the core. The router has infinite bandwidth and its task is to forward packets to appropriate peers. The star topology corresponds to a common assumption that bandwidth bottlenecks in P2P networks are at access links of peers to the Internet rather than in the core of the Internet [85]. Nevertheless, bandwidth bottlenecks in the core of the Internet do not affect the correct functioning of MeshTV. Bandwidth bottlenecks in the core of the Internet, typically caused by congestion, may reduce the available bandwidth between some peers. This may cause problems in P2P systems which correct functioning relies on estimating the uplink and downlink bandwidth of peers. Such P2P systems include ChunkySpread, PRIME and most of tree-based systems in which the peer out-degree is determined based on the peer uplink bandwidth. In these systems, two peers may connect to each other based on their uplink and downlink bandwidth, while a bandwidth bottleneck at an intermediate router may prevent transmission of data between these two peers at a required rate. In contrast, peers in MeshTV continuously refine their set of senders based on their download rate from these senders. When a bandwidth bottleneck in the core of the Internet reduces the download rate of a peer from its sender, the peer may decide to replace this sender with a new exploratory sender.

The goal of our network simulator is to simulate the transmission of packets over the network so that:

- Peers transmit data at the rate not exceeding their uplink bandwidth.
- Peers receive data at the rate not exceeding their downlink bandwidth

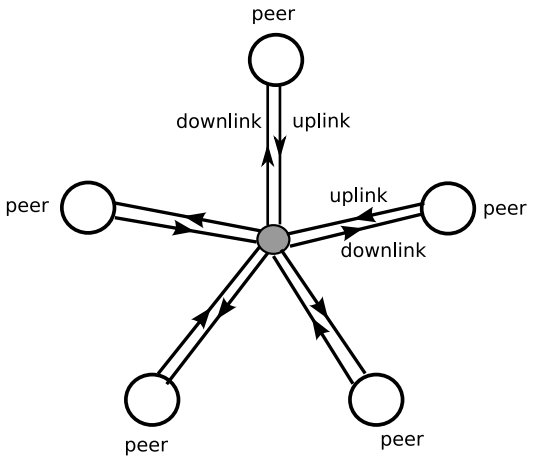


Figure 5.2: Star network topology.

- The bandwidth of a network link (i.e., uplink or downlink) is shared equally by connections that use this link.
- A portion of the link bandwidth allocated to a connection and unused by this connection is shared by remaining connections that use the same link so that the entire link bandwidth may be utilised.
- Network links impose delays on the propagation of packets.

Figure 5.3 illustrates the transmission of a packet from peer X to peer A in our network simulator. Initially, peer X creates a packet and adds a header of 40 bytes to the packet. The header of such size imitates the TCP/IP header. The maximum size of a packet including its header is 1500 bytes, which corresponds to the typical value of the Maximum Transmission Unit (MTU) for Ethernet links [115]. In step *1*, the packet is inserted at the end of the outgoing packet queue for the appropriate connection. An outgoing packet queue is a FIFO queue that is maintained at a peer for each outgoing connection to a remote peer. It is used to queue packets before they are transmitted over the uplink as only a single packet is transmitted at a time. The transmission of a packet over either uplink or downlink imposes a delay that is

calculated as *packetSize*/*linkBandwidth*, where *packetSize* is the size of the packet and *linkBandwidth* is the bandwidth of the link. In step *2*, a packet is selected for the transmission over the uplink. To select a packet for the transmission, an outgoing packet queue is first selected and then the first packet from this queue is selected. The outgoing packet queue is selected in a round-robin fashion, where consecutive packets are selected for the transmission from consecutive non-empty queues. This ensures that packets are transmitted by peer X at the total rate not exceeding the peer's uplink bandwidth and that each outgoing connection is allocated an equal share of the available uplink bandwidth. Moreover, when the number of packets sent over a connection is insufficient to use this connection's share of the uplink bandwidth, remaining connections share the spare bandwidth. Once the packet is transmitted over the peer uplink, it is moved in step *3* to a buffer that imposes a network propagation delay. When this delay expires, the packet is moved in step *4* to an appropriate incoming packet queue at the destination peer A. The packet waits in this queue for its turn to be selected for the transmission over the downlink of peer A in step *5*. The transmission of packets over the downlink is analogous to the described transmission of packets over the uplink. After the packet is transmitted over the downlink, it is delivered to peer A in step *6*.

5.3 Experimental Setup

P2P live streaming systems attract hundreds of thousands of simultaneous viewers [50]. As accurate simulation of such large numbers of peers is currently infeasible, researchers aim at maximising the number of simulated peers. In our experiments, we simulate 5000 peers. This is close to the maximum number of peers that can be simulated in our network simulator in a reasonable time using our hardware. Due to the high performance of our network simulator, this number exceeds the number of peers used in evaluations of many other mesh-based P2P live streaming systems

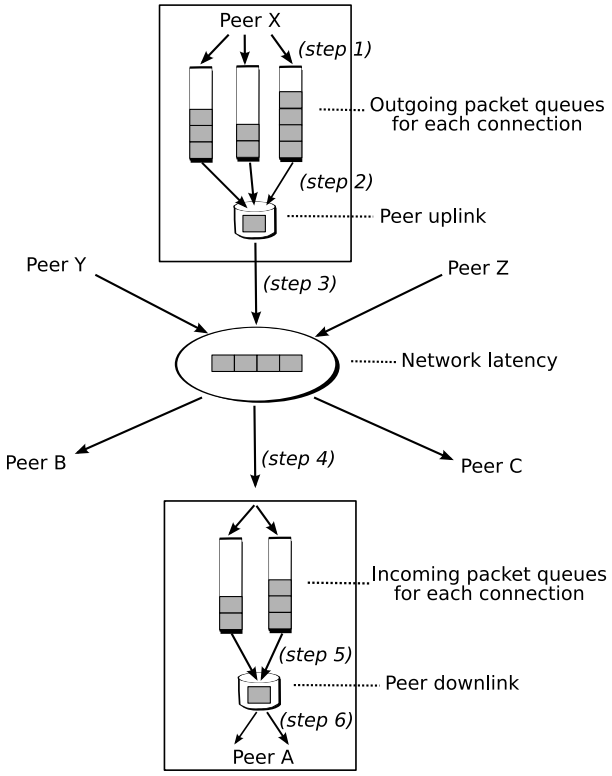


Figure 5.3: MeshTV network simulator.

[75, 90, 135]. To evaluate the time required to adapt the overlay as a function of the number of peers, we also run experiments with a larger number of peers, however, at reduced simulation accuracy as described in Section 5.4.5. Values of parameters of the MeshTV system are chosen experimentally and represent various tradeoffs that we discuss in this chapter. We experiment with stable overlays, where the set of participating peers or their bandwidth do not change over time, and with dynamic overlays, where we simulate peer churn and catastrophic failures.

Network setup. Table 5.1 shows the distribution of bandwidth among peers that is derived from measurements of the Gnutella P2P system [105], and so it approximates the distribution of peer bandwidth in the real world. In the distribution, peers are categorised into 4 groups: A, B, C and D, where each group has different downlink and

Category	Downlink	Uplink	Ratio of Peers
A	10 Mbps	5 Mbps	15%
B	3 Mbps	1 Mbps	25%
C	1.5 Mbps	384 Kbps	40%
D	784 Kbps	128 Kbps	20%

Table 5.1: Peer bandwidth distribution.

uplink bandwidth.

In experiments in Section 5.4, the uplink bandwidth of peers is set according to the given bandwidth distribution, however, the downlink bandwidth of peers is unlimited. We do not limit the downlink bandwidth in order to show that the overlay adapts so that the download rate of all peers approaches the maximum uniform download rate of 1179 Kbps. This maximum uniform download rate is the download rate that is uniform among peers and that maximises the uplink bandwidth of peers. It can be calculated with Equation 4.3.

In experiments in Section 5.5, the uplink and downlink bandwidth of peers are set according to the bandwidth distribution. The downlink bandwidth of peers is limited in these experiments in order to show that each peer adapts its quality of playback to its download rate that may be reduced by its individual downlink bandwidth.

In all experiments, a single transmitter peer is used to generate the media stream. The transmitter belongs to the category A with the uplink bandwidth of 5 Mbps.

Furthermore, in all experiments, a one-way network propagation delay between each two peers is selected randomly from 50ms, 100ms, 150ms and 200ms.

MeshTV parameters. Table 5.2 presents the summary of MeshTV parameters that are common for experiments in this chapter. Other parameters that are specific to particular experiments are discussed in sections dedicated to these experiments. Initially, we set parameters in Table 5.2 intuitively, based on tradeoffs discussed below and based

Parameter	Value
peer in-degree (K)	10
exploration round length	10 sec
chunk size (*ChunkSize*)	4 KB
max. requests pipelined to a sender (*max_pipelined*)	4
playback lag time (Δ)	30 sec
min. progress to increase the target quality level (γ)	10 sec
interval of periodic adaptation of the target quality level	5 sec

Table 5.2: MeshTV parameters common for experiments.

on values of similar parameters in other P2P systems [22, 90]. To adjust and verify these parameter settings, we subsequently ran, for each parameter, 4 to 6 experiments using different parameter values. We found that initial parameter settings produced desired and anticipated experimental results.

The parameter K denotes the number of senders (in-degree) of each peer. Its value of 10 has been selected experimentally and represents tradeoffs that we discuss in Section 5.4.4, 5.4.5, and 5.4.6. In Section 5.4.4 and 5.4.5, we also experiment with different values of K.

The exploration round length defines the time interval between consecutive executions of the exploration action at each peer. It represents a tradeoff between the speed of overlay adaptation and the communication overhead related to connecting and disconnecting a sender by a peer. Moreover, it should be sufficiently long to allow a peer to calculate the average download rate from each of its senders. BitTorrent uses a similar parameter to describe the length of the time interval between consecutive choke/unchoke operations performed by a peer during which the peer needs calculate the average download rate from each of its senders. We experimentally found that 10 seconds, which is also the value used by BitTorrent, is a sufficiently long interval to calculate the download rates and that 10 seconds enables fast overlay adaptation at a low communication overhead.

The size of a chunk represents a tradeoff between the communication overhead and the delay in the reception of the stream by peers. By increasing the chunk size, the number of NOTIFY and REQUEST messages sent by peers decreases, while the time of transmission of each chunk between two peers increases. We use the value of 4 KB that is the same as the one used in [90].

The *max_pipelined* parameter defines the upper limit on the maximum number of unsatisfied chunk requests that a peer can issue to a single sender. Once this limit is reached, the peer can issue a new chunk request to the sender only when the sender satisfies a previous request. The value of this limit should be sufficiently high to eliminate the idle time between consecutive transmission of chunks by a sender to a receiver. To eliminate this idle time, the value of *max_pipelined* should be at least equal to the number of chunks that can be transmitted by a sender to a receiver during the time of a packet round-trip between the sender and the receiver. This number can be calculated as: $DownloadRate * RTT/ChunkSize$, where $DownloadRate$ is the maximum download rate of the receiver from the sender, RTT is the round-trip time of a packet between these two peers and $ChunkSize$ is the size of a chunk. In turn, as we discuss in Section 5.4.4, the value of *max_pipelined* should not be too high as it increases the delay in the reception of chunks by peers. We experimentally found that the value of 4 eliminates the transmission idle time and allows for a low chunk reception delay at peers.

The playback lag time (Δ) defines the delay between the time when the media stream is produced by the transmitter and the playback time at peers. As we discuss in Section 5.4.4, the playback lag time of a peer must be higher than the delay in the reception of a media stream by the peer to allow for continuous playback. If the playback lag time is lower than the stream reception delay of a peer, some needed chunks may not be delivered to the peer before the playback time. This results in playback interruptions that typically increase the playback lag time so that it matches

the stream reception delay. In Section 5.4.4, we show that the playback lag time of 30 seconds is sufficiently high for uninterrupted playback at 5000 peers.

The minimum progress to increase the target quality level (γ) is a parameter used in Algorithm 1 and 3, described in Section 4.2.3 of this thesis. We experimentally found that its value of 10 seconds prevents the increase of the number of descriptions to download by a peer when this increase may result in oscillations in the quality of playback.

The last parameter in the table defines the time interval between periodic adaptations of $TargetQuality$ in Algorithm 1, described in Section 4.2.3. The length of this time interval must be sufficiently long to allow a peer to calculate the average aggregate download rate from senders. In turn, for timely adaptation to changing download rates, this interval should not be too long. We experimentally found that its value of 5 seconds represents a good tradeoff between these two requirements.

Initial overlay. In all experiments, the overlay is initially random, formed by peers selecting their senders at random. This represents a real-world situation called *flash crowd* [8], where a large population of users joins at the same time to obtain popular content when it becomes available. In live streaming, flash crowds often coincide with the beginning of live transmissions.

Peer churn. We experiment with a stable overlay, where the set of peers in the overlay does not change during an experiment, and with peer churn, where peers continuously join and leave the overlay. In experiments with peer churn, churn starts 500 seconds after the beginning of the experiments. The beginning of peer churn is delayed with respect to the beginning of the experiments in order to investigate the impact of peer churn on the already adapted overlay. To simulate peer churn, we use a methodology similar to [99, 71]. In this methodology, peer departures follow a Poisson process and thus are uncorrelated with each other. A new peer starts each time another peer

leaves the overlay, so the size of the overlay is constant. Churn rates are represented by a median duration of a peer session, t_{med}. The rate of peer arrivals can be calculated using the fact that inter-arrival times in a Poisson process with an arrival rate λ are exponentially distributed with a mean λ^{-1} and a median $ln2/\lambda$. It follows that an arrival rate of a single peer is $ln2/t_{med}$, whereas the rate of any arrival in the overlay with N peers is $N * ln2/t_{med}$. Therefore, a new peer arrives (and one departs) on average every $t_{med}/(N * ln2)$ seconds and these inter-arrival times are exponentially distributed.

We use median session durations of 30, 15 and 5 minutes, for which the mean inter-arrival times in an overlay with 5000 peers are 0.52, 0.26 and 0.09 seconds, respectively. These settings generate rates of peer churn even higher than those observed in real P2P systems, such as Gnutella and Napster, for which the measured median session duration was about 60 minutes [105].

Catastrophic failures. We also experiment with *catastrophic failures*, where 50% of all peers leave the overlay at the same time. This represents a real-world scenario where a large population of viewers is only interested in a single popular transmission and leaves the system when this transmission finishes. Departing peers may affect the topology of the overlay and thereby the delivery of content to peers remaining in the overlay.

5.4 Adaptation of Mesh Overlay

In this section, we analyse the adaptation of the mesh overlay by the enhanced MeshTV overlay adaptation algorithm presented in Section 4.3.4. In Section 5.4.1, we show that the overlay adapts so that the out-degree of every peer adapts to the peer's uplink bandwidth, the upload rate of every peer is maximised, and download rates are nearly uniform among peers. In Section 5.4.2, we investigate the resilience of

peer download rates to peer churn and show that even excessive peer churn reduces the average download rate of peers by only 5.5%. In Section 5.4.3, we investigate the resilience of download rates to catastrophic failures and show that a catastrophic failure temporarily reduces the average download rate by only 1.4%. In Section 5.4.4, we show that the delay at which the media stream is received by peers in the adapted overlay is uniform among peers and does not exceed 7 seconds for 5000 peers. Moreover, we analyse the influence of the peer in-degree (K) on this stream reception delay. In Section 5.4.5, we evaluate the time required to adapt the overlay from an initially random overlay for different peer in-degree (K) and for different number of participating peers (N). We show that this time is independent of the number of participating peers and is about 4 exploration rounds for $K = 10$. Finally, in Section 5.4.6, we analyse the communication overhead of MeshTV and show that it is about 13% of media content transmitted in the overlay.

5.4.1 Optimisation of Download Rates

Purpose and outcome. In this section, we analyse the adaptation of the mesh overlay and show that the out-degree of each peer adapts to the peer's uplink bandwidth, the upload rate of every peer is maximised, and download rates are nearly uniform among peers.

Experimental setup. In the experiments presented in this section, we simulate 5000 peers with the uplink bandwidth set according to the distribution in Table 5.1 and unlimited downlink bandwidth. An exception is the experiment presented in Figure 5.10, where we simulate peers with the downlink bandwidth limited according to the distribution in Table 5.1. This particular experiment is presented to show that the uplink bandwidth unused by peers with low downlink bandwidth increases the download rate of remaining peers.

To experimentally obtain the maximum download rate of peers, we set the rate of the media stream above the maximum uniform download rate of 1179 Kbps. We selected 1500 Kbps as this stream rate. In this section, we do not analyse the quality of playback and so we simulate a single media description.

The upload and download rate of each peer is calculated as the average over 10 second intervals and includes all data, respectively, uploaded or downloaded by the peer. These data include media content and the communication overhead, such as packet headers and MeshTV control messages. The communication overhead is analysed in Section 5.4.6. Furthermore, the upload and download rates in MeshTV are compared to those in Chainsaw. Chainsaw and MeshTV use a similar approach for chunk dissemination that is based on parallel downloads with pipelining, which allows for their comparison using the same chunk size, K, $max\_pipelined$ and Δ system parameters.

The experiments begin at time 0 seconds, when each peer selects 10 senders at random from all participating peers and when the transmitter starts producing and transmitting data chunks.

Analysis of peer out-degrees. Figure 5.4 shows the average and standard deviation of peer out-degrees, for each peer category, over time. The standard deviation is presented as error bars. The figure shows that initially every peer has, on average, 10 receivers. This is because the mesh overlay is formed by each peer selecting 10 senders at random. The out-degree of every peer adapts to the peer's uplink bandwidth and after about 8 exploration rounds, i.e., 80 seconds, the average peer out-degree stabilises in each peer category. The optimum out-degree of a peer can be calculated with Equation 4.6. For the uplink bandwidth distribution used in the experiments and $K = 10$, the optimum out-degree is roughly 42.4, 8.5, 3.3 and 1.1, respectively, for peers in categories A, B, C and D. The figure shows that the average peer out-degree converges to the optimum for each peer category. The figure also shows that the standard deviation of

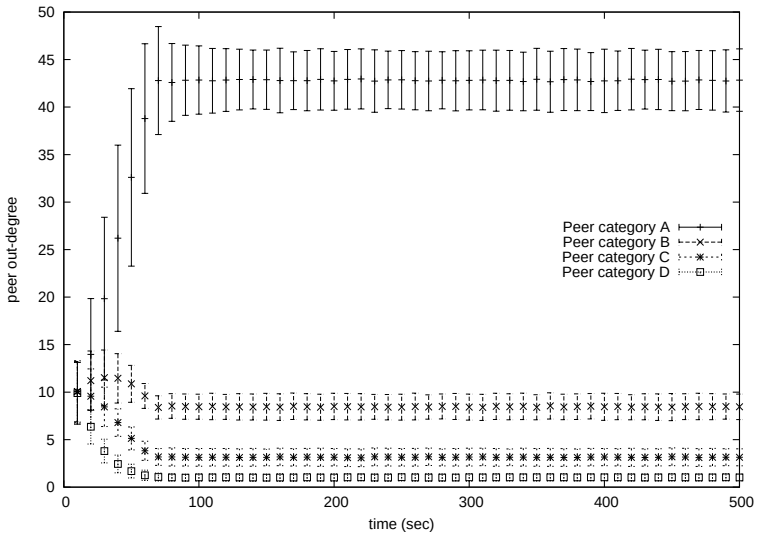


Figure 5.4: Average and standard deviation of peer out-degrees, for each peer category, over time.

peer out-degrees, once stabilises, is low and ranges from 3 for peer category A to 0.3 for peer category D.

Figure 5.5 presents the peer out-degree distribution. The distribution is computed at a time after the average out-degree stabilises. The figure is split into four subfigures, each presenting the distribution for a different peer category in order to allow for comparison of these distributions. The subfigures show that the variance of peer out-degrees increases with the uplink bandwidth of peers. This can be also observed in Figure 5.4 by comparing the standard deviation of peer out-degrees in different peer categories. The reason for this is following. A decrease in the out-degree of a peer is a consequence of a receiver disconnecting from this peer. In turn, an increase in the out-degree of a peer is a consequence of a receiver providing this peer as an exploratory sender to another peer. However, the decision about disconnecting a peer or providing a peer as an exploratory sender is probabilistic and may result in the variance of out-degrees among peers. This variance is higher among peers with higher uplink bandwidth, because the number of receivers that decide about increasing or decreasing

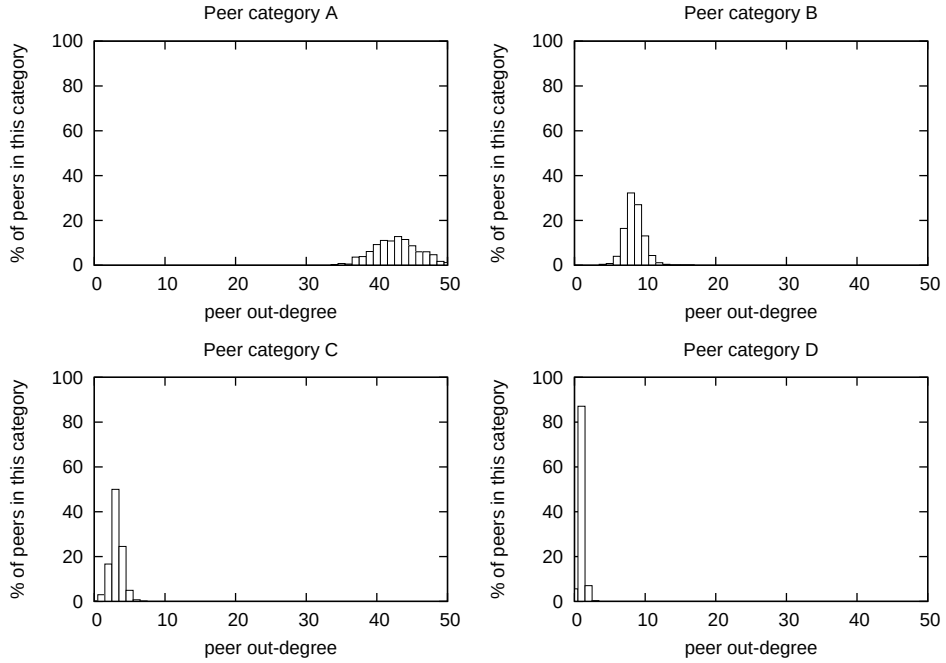
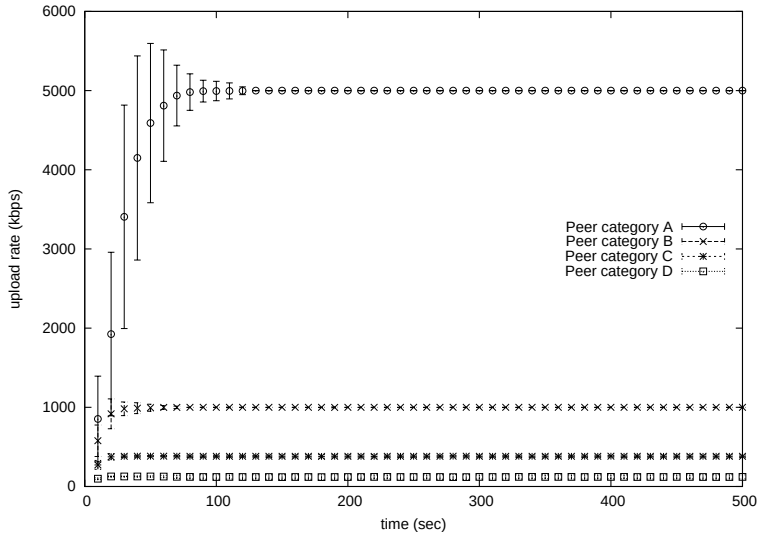


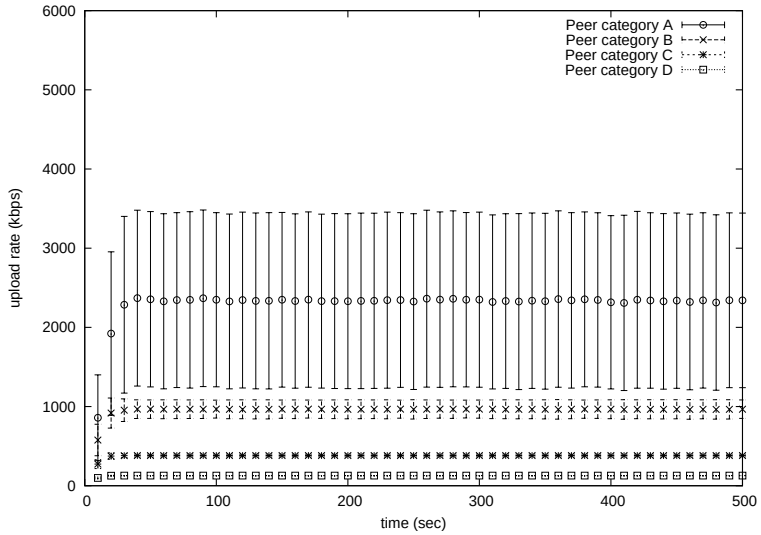
Figure 5.5: Peer out-degree distribution for each peer category.

the out-degree of a peer is proportional to the uplink bandwidth of the peer. However, the goal of overlay adaptation is to equalise the uplink-to-outdegree ratios among peers. Despite uneven variance of peer out-degrees, the variance of peer uplink-to-outdegree ratios is roughly the same among all peer categories. This is because the impact of the out-degree variance on the uplink-to-outdegree variance is inversely proportional to the uplink bandwidth.

Analysis of peer upload rates. Figures 5.6(a) and (b) show the average and standard deviation of peer upload rates, for each peer category, over time in MeshTV and in Chainsaw. The transmitter begins producing chunks at time 0 seconds, so initially peers have no chunks to upload to their receivers and thus the upload rate of peers is equal to 0. Peers download increasing amounts of chunks from their senders and upload these chunks to their receivers. Thus, the upload rate of peers increases until



(a) MeshTV



(b) Chainsaw

Figure 5.6: Average and standard deviation of peer upload rates over time in MeshTV and in Chainsaw.

it eventually stabilises. However, the bottom figure shows that Chainsaw underutilises the uplink bandwidth of peers with high uplink bandwidth. Peers in category A upload on average 2000 Kbps, while their available uplink bandwidth is 5000 Kbps. Moreover, the standard deviation shows a large variance in the upload rate of these peers. In contrast, the upload rate of each peer in MeshTV converges to the available uplink bandwidth of the peer.

Figure 5.7 presents the cumulative distribution function (CDF) of peer upload rates for MeshTV and Chainsaw. The distribution is computed at a time after peer upload rates stabilise. For rate x, the CDF shows the percentage of peers with the upload rate not greater than x. The figure shows that, in MeshTV, almost 20% of peers upload at 128 Kbps, almost 40% of peers upload at 384 Kbps, 25% of peers upload at 1000 Kbps and 15% of peers upload at 5000 Kbps. These numbers correspond to the distribution of the peer uplink bandwidth in Table 5.1, confirming that MeshTV utilises nearly the entire uplink bandwidth of all peers.

The figure shows that Chainsaw utilises the entire uplink bandwidth of only those peers that have low uplink bandwidth. The reason for this is that peers in Chainsaw do not adapt their out-degree. Peer out-degree of 10 is sufficient to use the entire uplink bandwidth of low capacity peers in categories C and D, however, it is too low to use the entire uplink bandwidth of high capacity peers in categories A and B. Therefore, much of the uplink bandwidth of peers is not utilised by Chainsaw.

Analysis of peer download rates. In previous experiments, we showed that MeshTV uses almost the entire uplink bandwidth of all peers by adapting the out-degree of each peer to the uplink bandwidth of the peer. Here, we show that download rates are nearly uniform among all peers and approach the maximum. Figure 5.8 shows the average and standard deviation of peer download rates over time in MeshTV and in Chainsaw. The figure also shows the download rate that is uniform among peers

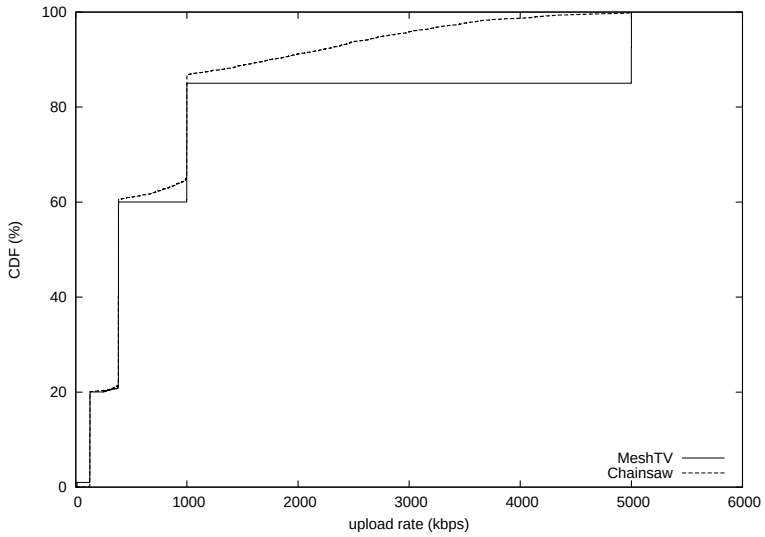


Figure 5.7: Peer upload rate distribution.

and maximum for the distribution of peer uplink bandwidth. This maximum uniform download rate is calculated with Equation 4.3 and is equal to 1179 Kbps. The figure shows that, in Chainsaw, the average peer download rate is only 765 Kbps and that the standard deviation is about 220 Kbps. In contrast, the average peer download rate in MeshTV approaches the maximum of 1179 Kbps and the standard deviation is only 55 Kbps.

Figure 5.9 presents the CDF of peer download rates for MeshTV and Chainsaw. The distribution is computed at a time after peer download rates stabilise. It shows that peer download rates in Chainsaw are non-uniform, ranging from 200 Kbps to 1300 Kbps. Such low and non-uniform download rates do not allow for high quality playback at peers. In contrast, peer download rates in MeshTV are nearly uniform and range from 1000 Kbps to 1400 Kbps. Such high download rates allow for continuous playback at high quality even when a single media description is used.

Finally, Figure 5.10 compares two CDFs of peer download rates in MeshTV: one for the limited peer downlink bandwidth and one for the unlimited downlink bandwidth.

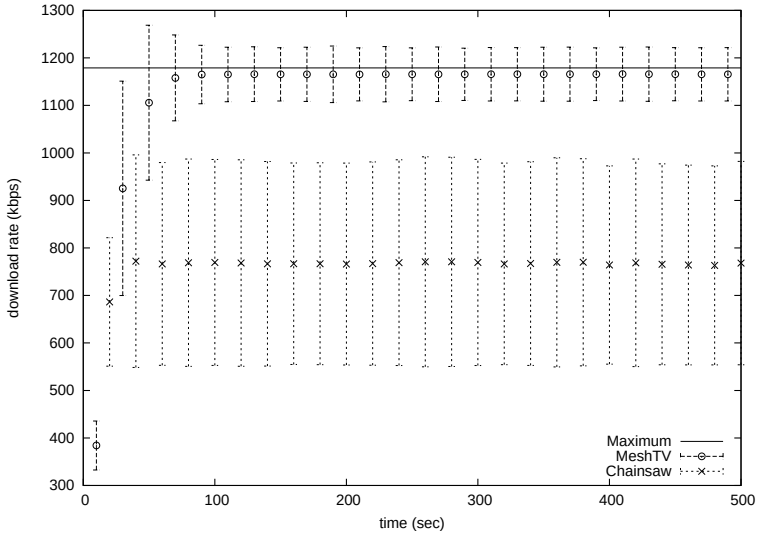


Figure 5.8: Average and standard deviation of peer download rates in MeshTV and in Chainsaw.

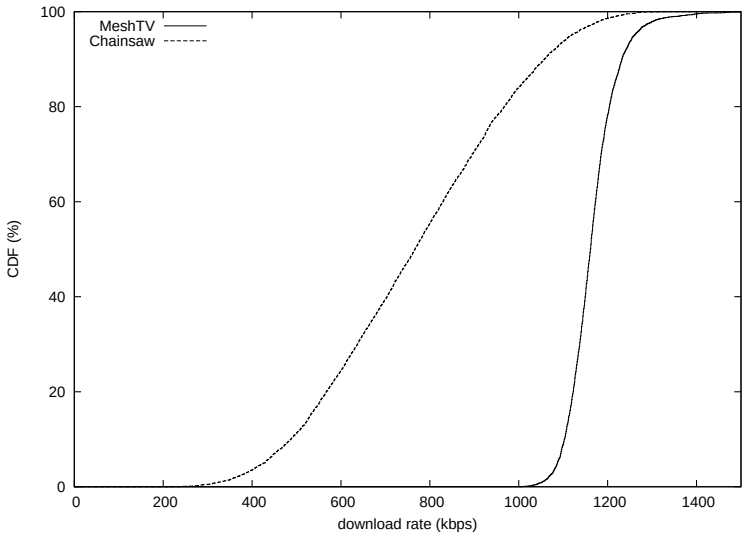


Figure 5.9: Peer download rate distribution.

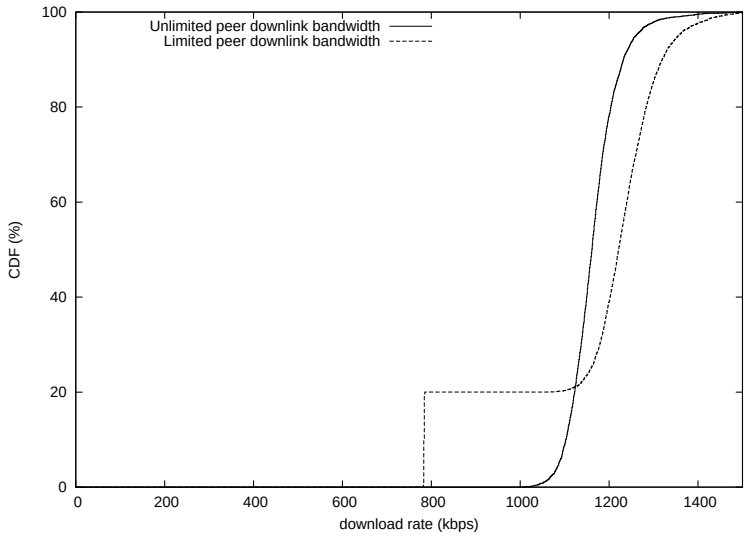


Figure 5.10: Peer download rate distribution for limited and unlimited downlink bandwidth.

The figure shows that when the download rate of some peers is reduced by the downlink bandwidth of these peers, the uplink bandwidth unused by these peers increases the download rate of remaining peers.

5.4.2 Resilience to Peer Churn

Purpose and outcome. In this section, we evaluate the impact of arriving and departing peers on the download rate of remaining peers. Experimental results show that even excessive peer churn with the median peer session time of 5 minutes reduces the average download rate by only 5.5%.

Experimental setup. Experiments in this section use the same settings as those presented in Section 5.4.1 and additionally simulate peer churn. To evaluate peer churn in the already adapted overlay, we selected the time of the beginning of peer churn as 500 seconds. To show the impact of arriving and departing peers on the download rate of remaining peers, only peers that are present in the overlay for over 60 seconds

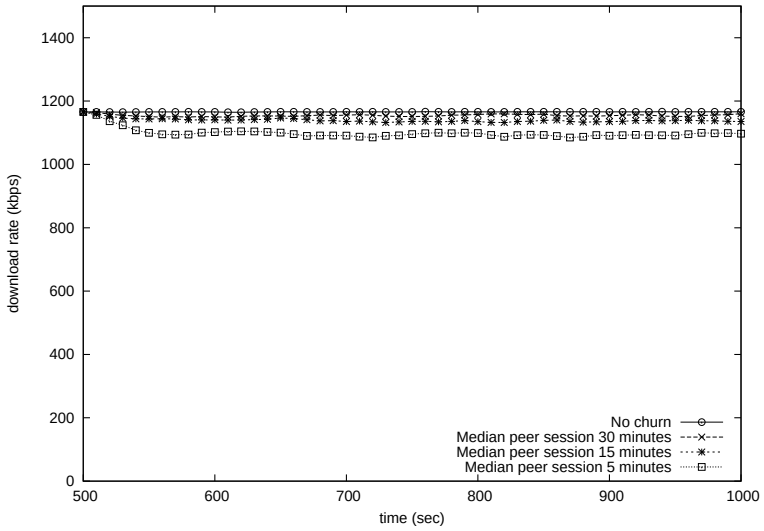


Figure 5.11: Average peer download rate over time for different rates of peer churn.

are included in the computed average and standard deviation. We experiment with different rates of peer churn, represented by median peer session times of 30, 15, and 5 minutes. For comparison, we also show results of experiments without peer churn.

Analysis. Figure 5.11 shows the average and standard deviation of peer download rates over time for different rates of peer churn. The figure shows that even under excessive rates of peer churn, the average download rate remains high. In particular, the median peer session time of 5 minutes corresponds to the mean interval of 0.09 seconds between consecutive peer departures in the overlay with 5000 peers. For such excessive churn rate, the average download rate decreases by only about 5.5% compared to the average download rate in the experiment without peer churn.

Figure 5.12 presents the peer download rate distribution for different rates of peer churn. The figure shows that, in addition to reducing the average download rate, peer churn increases the variance of peer download rates. In particular, the standard deviation of peer download rates is 117 Kbps, 80 Kbps, 70 Kbps, and 55 Kbps, respectively,

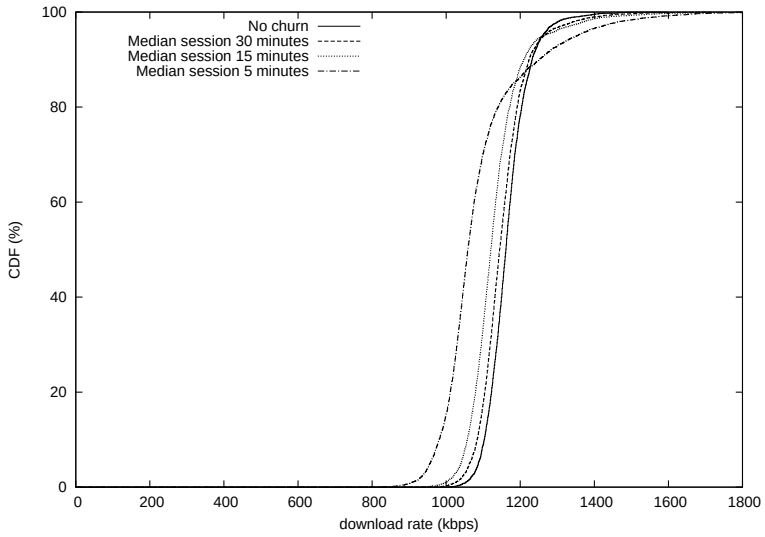


Figure 5.12: Peer download rate distribution for different rates of peer churn.

for the experiments with median peer session time of 5 minutes, 15 minutes, 30 minutes, and for the experiment without peer churn.

5.4.3 Resilience to Catastrophic Failures

Purpose and outcome. In this section, we evaluate the impact of a catastrophic failure, where 50% of all peers fail at the same time, on the download rate of peers remaining in the overlay. Experimental results show that the failure temporarily reduces the average peer download rate by only 1.4% and that only 3 exploration rounds after the failure peer download rates become the same as those before the failure.

Experimental setup. Experiments in this section use the same settings as those presented in Section 5.4.1 and additionally simulate the catastrophic failure. To simulate the failure in the already adapted overlay, we selected the time of the failure as 500 seconds. Peers departing at the time of the failure consist of 50% of peers from each peer category so that the peer bandwidth distribution and the maximum uniform

download rate of 1179 Kbps do not change after the failure.

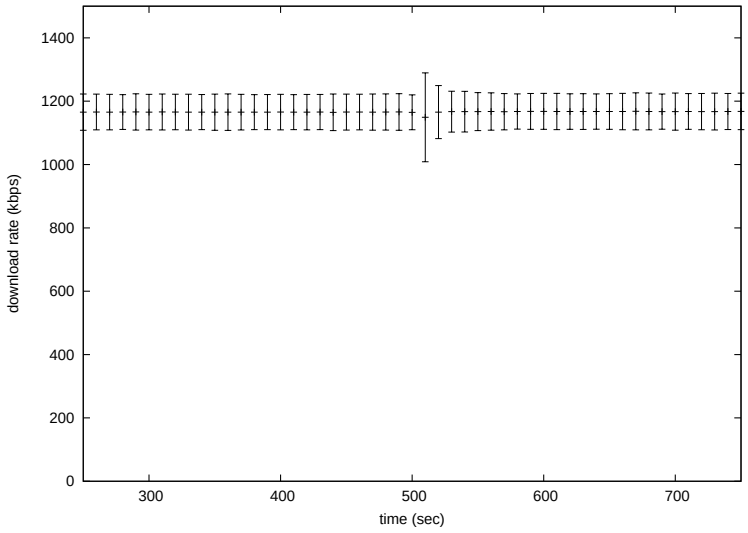


Figure 5.13: Average and standard deviation of peer download rates over time. 50% of peers fail at time 500 seconds.

Analysis. Figure 5.13 shows the average and standard deviation of peer download rates over time. The figure shows that 10 seconds after the catastrophic failure, the average download rate is lower by only 1.4% compared to the average download rate before the failure. The little impact of the failure on download rates can be explained as follows. When half of all peers depart, each of the remaining peers loses on average half of its senders and half of its receivers. Thus, a peer can upload content twice as fast as before the failure to each of its receivers that remained in the overlay. As a consequence, despite having half of their senders, peers download content from each sender twice as fast as before, so their download rate remains roughly the same. In only 30 seconds, the overlay adapts so that peers have the same number of senders and receivers as before the catastrophic failure.

The figure shows that the catastrophic failure has a larger impact on the standard deviation of download rates than on their average. This is because of the randomness

in the selection of departing peers. This randomness causes that some peers lose more senders or lose a larger proportion of receivers compared to other peers. Thus, the download rate of peers may decrease or increase by amounts that vary among peers. This causes the variance of download rates, but may not increase the average of these download rates.

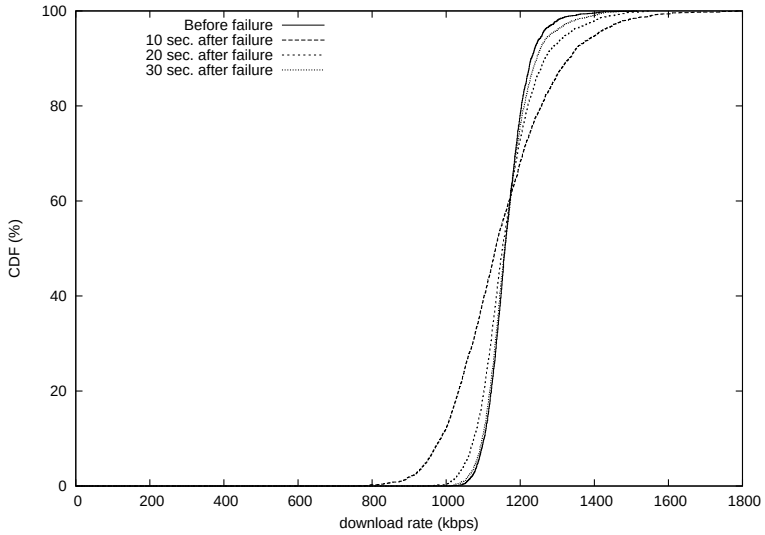


Figure 5.14: Peer download rate distribution before and after the catastrophic failure.

Finally, Figure 5.14 shows the peer download rate distribution before as well as 10, 20 and 30 seconds after the failure. It shows that 10 seconds after the failure the download rate of peers ranges from 800 Kbps to 1600 Kbps, however, only 20 seconds later the variance is approximately the same as before the failure.

5.4.4 Stream Reception Delay

To avoid playback interruptions and to allow for a low playback lag time, it is desirable to minimise the delay between the time when the media stream is produced by the transmitter and the time when it is received by a peer. This delay is called the *stream reception delay* of a peer.

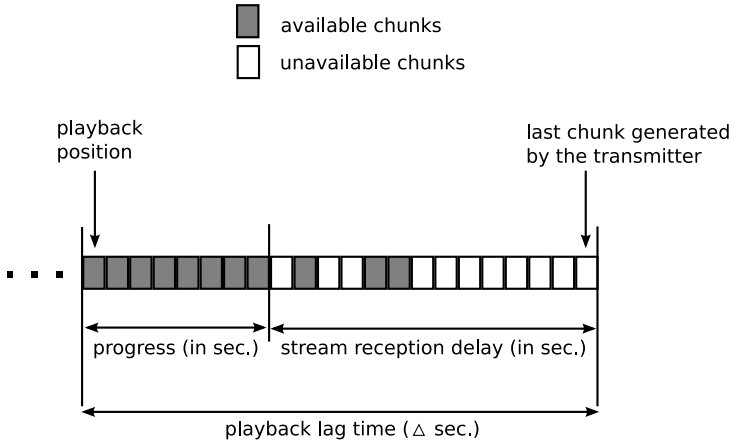


Figure 5.15: Stream reception delay determined from the buffer map of a peer (for $M = 1$).

In mesh-based approaches, the stream is composed of chunks that are delivered in a non-sequential order to peers. However, reconstruction of the stream for playback requires that peers possess consecutive chunks. Therefore, we define the stream reception delay of a peer as the delay between the time when chunks are produced by the transmitter and the time when these chunks are delivered in-sequence to the peer. Figure 5.15 illustrates how the stream reception delay of a peer can be determined using the buffer map of the peer, when a single media description is used. The figure shows that the stream reception delay of a peer is equal to the difference between the playback lag time of the peer and the progress of the peer, where the progress is equal to the length (in seconds) of the longest consecutive segment of chunks starting at the current playback position of the peer.

The playback lag time of a peer must not be lower than the stream reception delay of the peer. If the stream reception delay is higher than the playback lag time, some chunks are not delivered to the peer before their playback time. A chunk undelivered before its playback time results in a playback interruption. At the time of such interruption, the progress of the peer is equal to 0 and the stream reception delay is equal

to the playback lag time. The playback is then typically delayed until the missing chunk is delivered. This delay increases the playback lag time such that it matches the stream reception delay.

The stream reception delay of a peer is caused by chunks being forwarded over multiple peers. Each transmission of a chunk between two peers increases this delay. The main reasons for this are following. First, the propagation delay of a network link incurs a delay in the transmission of a data packet over the link. Second, the bandwidth of a network link is limited. To avoid packets being lost, packets are queued before they are transmitted over a network link. This packet queueing causes a delay. Finally, MeshTV uses a pipelining technique that incurs a delay. In this technique, a peer issues multiple requests for different chunks to a single sender and these chunks are transmitted in the order of requests. Thus, the delay incurred by the pipelining technique corresponds to the number of requests pipelined to a single sender and this number is limited by the value of $max\_pipelined$ parameter.

Purpose and outcome. In this section, we analyse the influence of two parameters, the stream rate and the peer in-degree K, on the stream reception delay of peers in MeshTV. For different values of these parameters, we show that the stream reception delay is low and uniform among peers.

Experimental setup. Experiments in this section use the same settings as those presented in Section 5.4.1 with the only difference that the rate of the media stream and the value of K are varied. To allow for continuous playback at peers, we reduce the stream rate below the maximum uniform peer download rate of 1179 Kbps. To analyse the influence of the stream rate on the stream reception delay, we experiment with stream rates of 800 Kbps and 1 Mbps. The value of K is set to 10 in these experiments. For comparison, we also run experiments with Chainsaw under the same settings.

To analyse the influence of parameter K on the stream reception delay, we run experiments with K set to 10, 15 and 20. The stream rate is set to 1 Mbps in these experiments.

Analysis of the stream reception delay for different stream rates. Figure 5.16 shows the distribution of the stream reception delay among peers in MeshTV and in Chainsaw for different stream rates. The figure shows that, in Chainsaw, the stream reception delay of most peers equals to their playback lag time, which is fixed at 30 seconds. This is because the download rate of these peers is below the stream rate, and so continuous playback is not possible at these peers. In contrast, the figure shows that the stream reception delay in MeshTV is below 7 seconds at all peers for each stream rate. The stream reception delay in MeshTV is nearly uniform among peers, meaning that the stream is delivered to all peers at approximately the same delay. The figure also shows that the stream reception delay decreases when the rate of the stream is reduced. When the stream rate is reduced, a peer can upload each chunk to more of its receivers, and thereby the dissemination time of chunks in the overlay decreases.

Analysis of the stream reception delay for different K. To investigate the influence of K on the stream reception delay of a peer, the adapted mesh overlay may be represented as K multicast trees [13, 76]. In this representation, each of K senders of a peer in the mesh overlay is a parent of this peer in one of the trees, as is illustrated in Figure 4.3. When K increases, the number of trees increases and the height of each tree decreases. Thus, increasing K reduces the length of the overlay path from the transmitter to each peer. However, increasing K increases the number of children of each peer. In turn, increased number of children increases the transmission time of a chunk between a parent and a child. Therefore, K represents a tradeoff between the length of overlay paths and the time it takes to transmit a chunk between two peers. In [13], we analytically showed that the value of K that minimises the stream reception

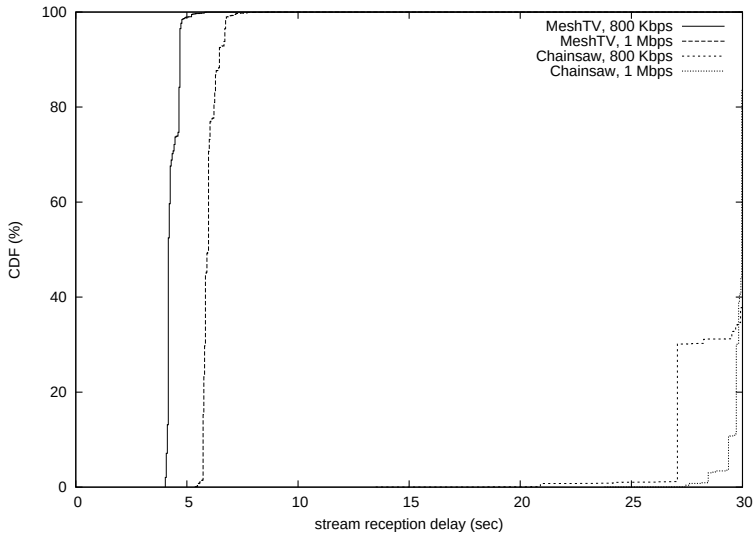


Figure 5.16: Distribution of the stream reception delay among peers in MeshTV and Chainsaw for different stream rates ($K = 10$).

delay depends on the distribution of peer bandwidth.

We experiment with different values of K to show its influence on the stream reception delay under the peer uplink bandwidth distribution from Table 5.1. The evaluation results are presented in Figure 5.17. The figure shows that the stream reception delay is low, nearly uniform among all peers, and increases with K.

5.4.5 Overlay Adaptation Time

Purpose and outcome. In this section, we evaluate the time required to adapt the mesh overlay using the enhanced algorithm described in Section 4.3.4. We show experimental results for different number of peers N and for different peer in-degree K. These results show that the adaptation time does not depend on N, and so the algorithm scales to any number of peers. The adaptation time depends on K, but it is low even for large K. For example, for K set to 10, the overlay adapts in only 4 exploration rounds.

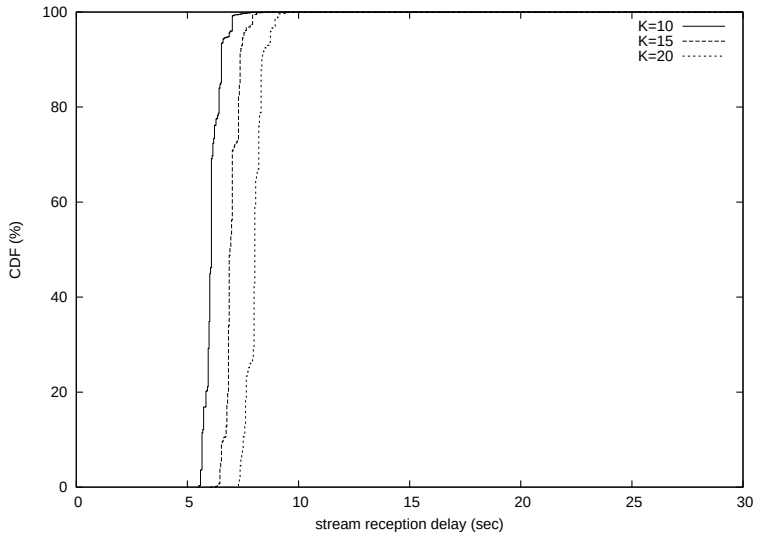


Figure 5.17: Distribution of the stream reception delay among peers for different K (and stream rate of 1 Mbps).

Experimental setup. To evaluate the time required to adapt the overlay as a function of the number of peers, we had to simulate much more than 5000 peers. Therefore, we built a new simulator. In contrast to our main simulator described in Section 5.2, this simulator is flow-level, meaning that it simulates transmission of batch data rather than transmission of individual data packets. Moreover, the simulator assumes that peers always possess chunks needed by their receivers, and so peers continuously upload chunks to their receivers. These simplifications reduce the complexity of the simulator, thereby allowing for experiments with up to 100 000 peers.

In the experiments in this section, the uplink bandwidth of peers is limited according to the uplink bandwidth distribution in Table 5.1, while the downlink bandwidth of peers is unlimited. The set of participating peers or their bandwidth do not change over time.

We first set N to 100 000 and experiment with different values of K to analyse the dependence between the value of K and the overlay adaptation time. Then, we set K to 10 and experiment with different values of N to analyse the dependence between

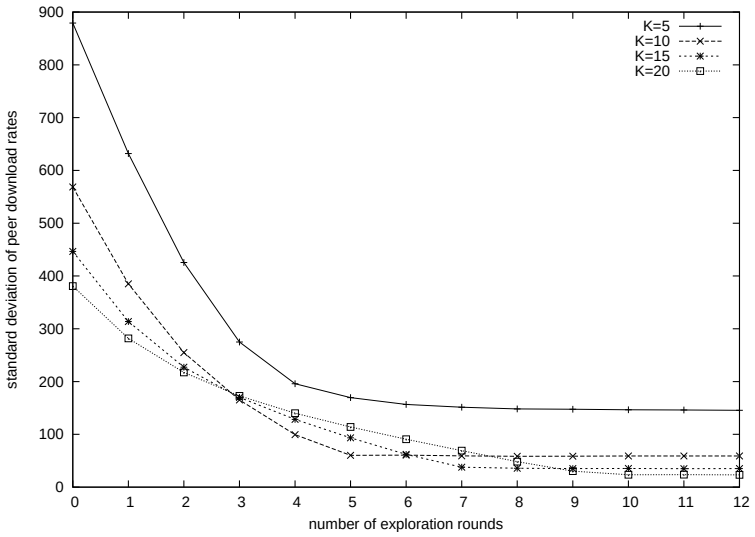


Figure 5.18: Convergence of peer download rates for different K ($N = 100000$).

the number of peers and the overlay adaptation time.

Analysis of the adaptation time for different values of K. Figure 5.18 presents the standard deviation of peer download rates over time for different values of K and $N = 100000$. Initially, the overlay is random and thus peer download rates have a high standard deviation. The reason for the variance in the download rate of peers is the variance in their download rate from each sender. The download rate from each sender may be different, because senders have heterogeneous uplink bandwidth, whereas their number of receivers is, on average, the same. The figure shows that in a random mesh overlay, i.e., at the exploration round 0, the standard deviation decreases when K increases. The reason for this is following. The download rate of a peer is the sum of its download rate from all senders. When the number of senders of a peer increases, the download rate of the peer approaches the average peer download rate according to the law of large numbers [31].

The figure shows that the standard deviation decreases as the overlay adapts. Ho-

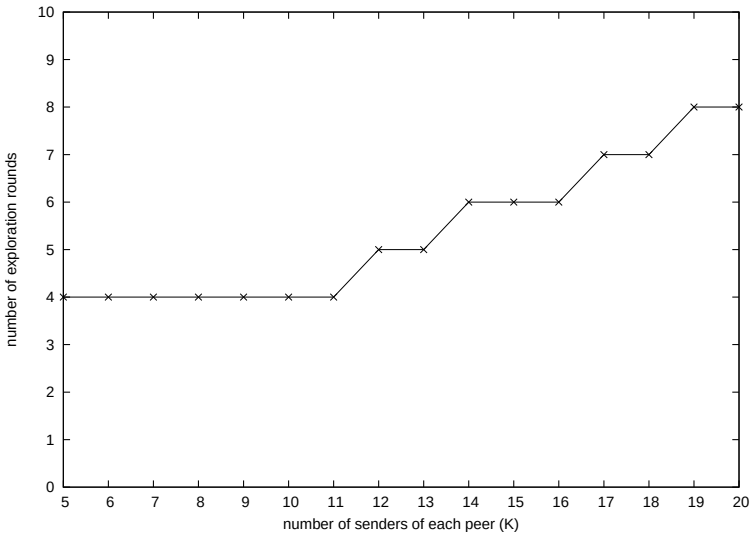


Figure 5.19: Number of exploration rounds required to adapt an initially random overlay as a function of K (for $N = 100000$).

wever, the standard deviation does not converge to 0. This is because the algorithm adapts the overlay continually to avoid convergence to a local optimum and to adapt to peer churn and to variations in the peer bandwidth. The figure shows that the minimum standard deviation is inversely proportional to K. This is because replacing a single sender may change the download rate to an extent that is inversely proportional to the number of senders.

Figure 5.19 shows the time required to adapt an initially random overlay as a function of K for $N = 100000$. For different values of K, it shows the number of exploration rounds required to reduce the standard deviation of peer download rates below the threshold given by $(\sum_{i=0}^{N} uplink_i)/(N * K)$. This threshold is calculated as the download rate from a single exploratory sender in the adapted overlay and thus corresponds to the discussed minimum standard deviation. The figure shows that the time of adaptation is low even for large K. For example, when $K = 20$, the out-degree of peers with the highest uplink bandwidth is 84, according to Equation 4.6, whereas the time of adaptation of peer out-degrees is only 8 exploration rounds.

Analysis of the adaptation time for different values of N. Figure 5.20 shows the relationship between the number of peers N and the time of adaptation, for $K = 10$. For different values of N, it shows the number of exploration rounds required to reduce the standard deviation of peer download rates below the threshold given by $(\sum_{i=0}^{N} uplink_i)/(N*K)$. For $K = 10$ and the uplink bandwidth distribution used in the experiments, the value of this threshold is 117.9 Kbps. The figure shows that the time of such adaptation does not depend on the number of participating peers. This is because overlay adaptation in MeshTV does not aim at connecting specific peers with each other. When specific peers need to be connected, searching in the overlay is typically required and the time of such searching depends on the number of peers participating in the overlay. In contrast, overlay adaptation in MeshTV allows for connections between any two peers and only aims at adapting the number of receivers of each peer. When a peer is underloaded, it gains new random receivers in each exploration round. When a peer is overloaded, it loses random receivers in each exploration round. The time of overlay adaptation in MeshTV depends on the peer in-degree K and the distribution of peer bandwidth.

5.4.6 Communication Overhead

In this section, we analyse the communication overhead incurred by the MeshTV protocol. This overhead is incurred by:

- CYCLON membership management protocol discussed in Section 4.3.5.
- Control messages, discussed in Section 4.2.2, related to the dissemination of chunks.
- MeshTV overlay adaptation algorithm discussed in Section 4.3.4.

The overhead of the CYCLON membership management protocol and the MeshTV overlay adaptation algorithm is small, especially when compared to the large volume

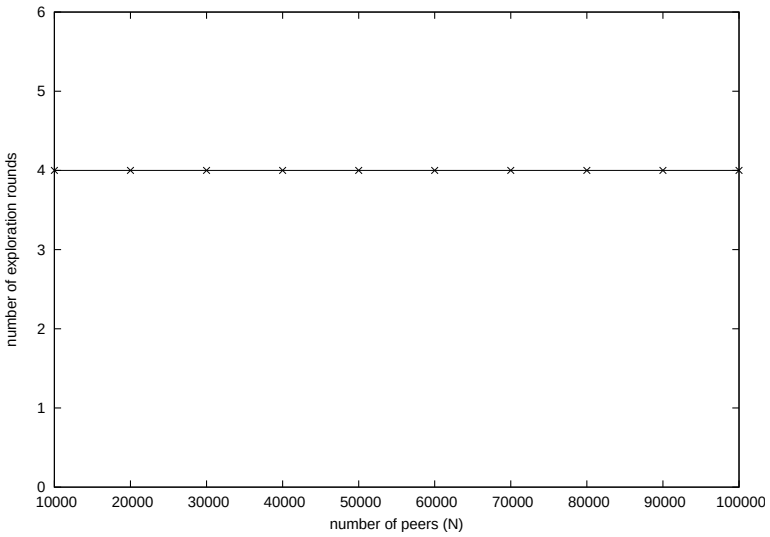


Figure 5.20: Number of exploration rounds required to adapt an initially random overlay as a function of N (for $K = 10$).

of media content transmitted by peers. In particular, in CYCLON, each peer sends on average two messages in each interval between consecutive executions of the shuffle algorithm. The length of this interval is 10 seconds. A peer sends one message to a peer selected at random from its partial view when it initiates the shuffle algorithm. The peer sends another message as a reply to a message received from another peer that initiated the shuffle algorithm. In turn, the enhanced MeshTV overlay adaptation algorithm requires that each peer periodically, every 10 seconds, performs the following actions. The peer disconnects one of its senders, requests an exploratory sender from another peer, connects to this exploratory sender, and may provide one of its senders to a requesting peer. Each of these actions require transmission of a single message, so the overlay adaptation algorithm does not incur much overhead.

Control messages related to the dissemination of chunks include: BUFFER MAP, NOTIFY, REQUEST and CHUNK messages. BUFFER MAP messages are sent sporadically, when two peers connect to each other, and thus do not incur much overhead. The CHUNK message consists of the message header and media content. The mes-

sage header contains only a description number and a timestamp, so it does not incur much overhead either. The largest overhead is incurred by REQUEST and NOTIFY messages. The REQUEST message is sent by a peer to request a chunk from a sender. The NOTIFY message is sent by a peer to notify each receiver about a new chunk available for download[1].

Here, we study the overhead of REQUEST and NOTIFY messages. If we assume that a peer downloads content at the rate of *DownloadRate*, then it downloads new chunks at the approximate rate of

$$\frac{DownloadRate}{ChunkSize}$$

where *ChunkSize* is the size of each chunk. This also represents the rate at which a peer sends REQUEST messages as a peer needs to request a chunk to download it. In turn, whenever a peer downloads a new chunk, it notifies all its receivers. Thus, the rate at which a peer sends NOTIFY messages is approximately

$$outdegree * \frac{DownloadRate}{ChunkSize}$$

where *outdegree* is the number of receivers of the peer. If we assume unlimited downlink bandwidth, then the download rate of all peers is approximately the same in the adapted overlay. As the sum of all out-degrees is equal to $N * K$, the rate at which any of N peers sends the REQUEST or the NOTIFY message is approximately

$$N * DownloadRate * \frac{K+1}{ChunkSize}$$

where K is the number of senders of each peer. In turn, the rate at which media content is sent in the overlay is approximately $N * DownloadRate$. Thus, REQUEST and NOTIFY messages represent $MS * (K+1)/ChunkSize$ of the media content sent

[1]As discussed in Section 4.2.2, a receiver of the transmitter also sends the NOTIFY message to notify the transmitter whenever it requests a chunk from another sender. In turn, it does not send REQUEST messages to the transmitter. However, due to the relatively small number of peers that are receivers of the transmitter, we omit these special cases from our discussion.

in the overlay, where MS is the size of each of these messages. In our experiments, we set MS to 48 bytes that is the sum of 40 bytes for the Internet protocol header (e.g., the length of the TCP/IP header is 40 bytes), 4 bytes for the description number field and 4 bytes for the timestamp field. Thus, for $MS = 48$ bytes, $K = 10$ and $ChunkSize = 4$ KB, the combined communication overhead of REQUEST and NOTIFY messages is about 13% of the whole media content transmitted in the overlay.

The communication overhead is proportional to $K + 1$ and inversely proportional to $ChunkSize$, so it could be reduced by decreasing K or increasing $ChunkSize$. However, when K decreases, the resilience of the download rate of peers to departure of their senders decreases. In turn, when $ChunkSize$ increases, the transmission time of a chunk between two peers increases and, as a consequence, the stream reception delay increases at peers.

5.5 Adaptation of Playback Quality

In previous sections, we showed that MeshTV adapts the overlay so that download rates are nearly uniform among peers and the upload rate of peers is maximised. However, we assumed unlimited downlink bandwidth of peers, whereas it may happen that limited downlink bandwidth reduces the download rate of some peers. Moreover, in the previous section, we showed that some variations in download rates may occur as a result of peer churn and catastrophic failures. Thus, to deliver continuous playback at high quality to peers with non-uniform download rates, the quality of playback at each peer needs to adapt to the peer's download rate.

In Section 5.5.1, we show that the quality of playback at each peer adapts to the peer's download rate. In Section 5.5.2, we show that peer churn reduces the average quality of playback by only about 5%. In Section 5.5.3, we show that the catastrophic failure temporarily reduces the average quality of playback by only about 5%, but

after only 30 seconds, the quality of playback becomes the same as before the failure. Finally, in Section 5.5.4, we show that playback of joining peers starts after only 3 seconds delay.

5.5.1 Optimisation of Playback Quality

Purpose and outcome. We show that the quality of playback at each peer adapts to the download rate that may be reduced by the limited downlink bandwidth. To quantify the quality of playback, we define the *level of playback quality* at a peer as the number of distinct chunks with the same timestamp t available in the peer's buffer, where t is the timestamp of the portion of the stream that is currently played. This level may range between 0, when no playback is possible, and M, when playback at the highest quality is possible. To avoid small oscillations of this level, we take its minimum over 1 second intervals.

Experimental setup. To experiment with non-uniform download rates, we limit the downlink bandwidth and the uplink bandwidth according to the bandwidth distribution in Table 5.1. The media stream is produced at the rate of 1500 Kbps and consists of 10 media descriptions, i.e., $M = 10$, so the rate of each description is 150 Kbps. The rate of 1500 Kbps is higher than the download rate of every peer in the adapted overlay so that the playback quality can be maximised at every peer. The number of descriptions represents a tradeoff between the fine granularity of adaptation of playback quality and the compression efficiency. By increasing the number of descriptions, the quality of playback at a peer may closer match the peer's download rate, however, the compression efficiency of MDC degrades.

Analysis. Figure 5.21 shows the average and standard deviation of the playback quality level of peers over time. It shows two plots: one for peers in categories A, B

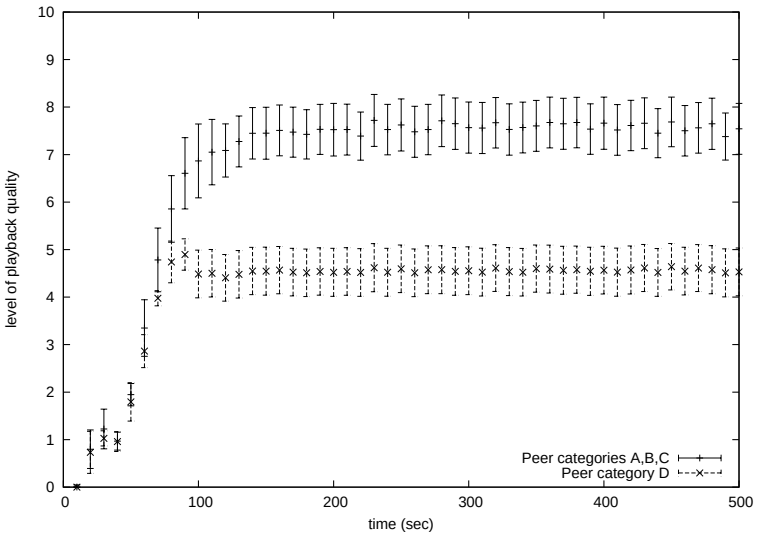


Figure 5.21: Average and standard deviation of the playback quality level of peers over time.

and C, and one for peers in category D. Peers in category D are separated from the remaining peers, because their low downlink bandwidth reduces their download rate. In contrast, the downlink bandwidth of peers in categories A, B and C does not reduce their download rate. The figure shows that the playback quality of peers adapts in 140 seconds. The average playback quality level stabilises at about 4.6 for peers in category D. As each description has the rate of 150 Kbps, the level of 4.6 corresponds to the download rate of 690 Kbps. This download rate is lower than the downlink bandwidth of 784 Kbps, because it excludes the communication overhead of packet headers and control messages. The figure also shows that the average playback quality level is about 7.6 for peers in categories A, B and C. Furthermore, the standard deviation is only 0.5.

Figure 5.22 shows the distribution of the playback quality level of peers in the adapted overlay. The distribution is presented as a bar chart. For each playback quality level, a bar shows the percentage of all peers with this playback quality level. The figure shows that the playback quality level of 20% of all peers is 4 or 5. These 20% of peers correspond to peers in category D. The playback quality level of the

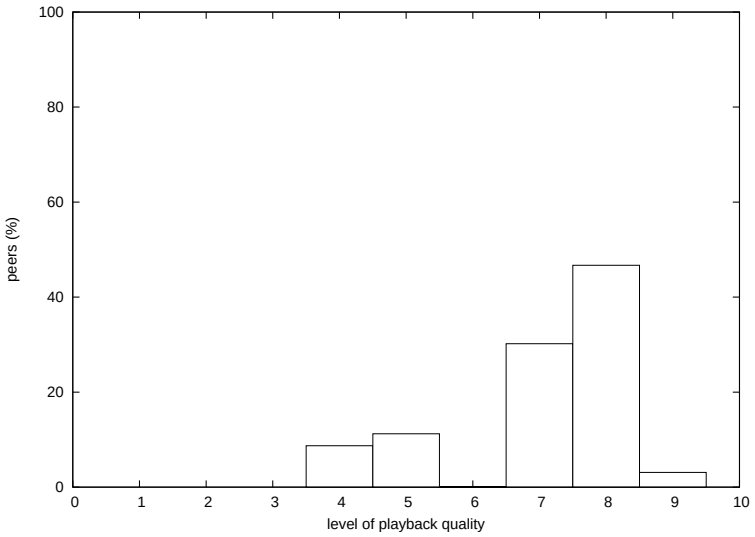


Figure 5.22: Distribution of the playback quality level of peers.

most remaining peers is 7 or 8. These results confirm that the quality of playback is nearly uniform among peers. In particular, all peers in category D deliver playback at a similar quality. Likewise, all peers in categories A, B and C deliver playback at a similar quality.

5.5.2 Resilience to Peer Churn

Purpose and outcome. In this section, we evaluate the impact of arriving and departing peers on the playback quality level of remaining peers. Experimental results show that peer churn with the median peer session time of 15 minutes reduces the average playback quality level of high capacity peers by only about 5%.

Experimental setup. Experiments in this section use the same settings as those presented in Section 5.5.1 and additionally simulate peer churn. The methodology for simulating peer churn is discussed in Sections 5.3 and 5.4.2. Peer churn begins at time 500 seconds and the median peer session time is 15 minutes. For comparison, we

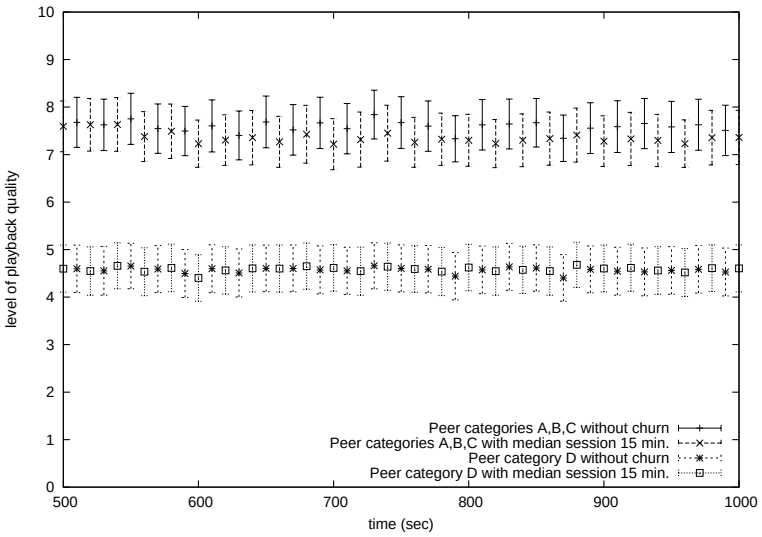


Figure 5.23: Average and standard deviation of the playback quality level of peers over time, with and without peer churn.

also show results for experiments that use the same settings, but do not simulate peer churn.

Analysis. Figure 5.23 shows the average and standard deviation of the playback quality level of peers over time for experiments with and without peer churn. The figure shows no impact of peer churn on the average or standard deviation of the playback quality level of peers in category D. This is because the download rate of these peers is reduced by their downlink bandwidth more than by peer churn. Peer churn reduces by about 5% the average playback quality level of peers in categories A, B and C. This is because peer churn reduces the download rate of these peers, as shown in Section 5.4.2. Finally, the figure shows no impact of peer churn on the variance of the playback quality among peers.

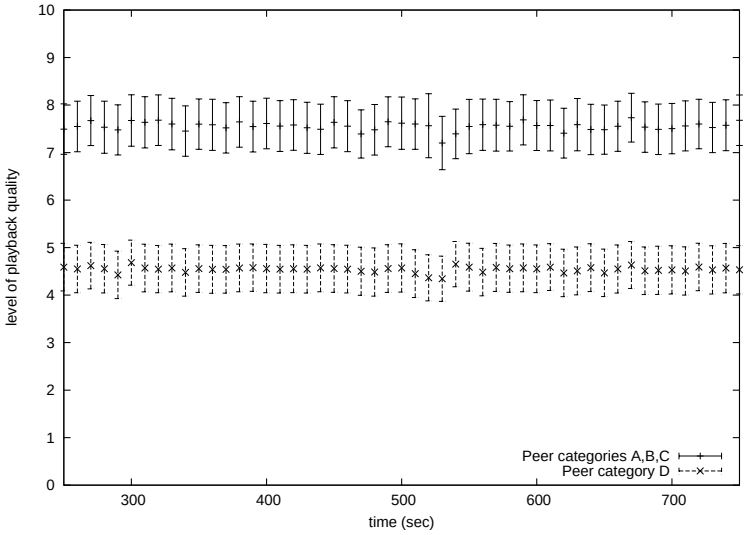


Figure 5.24: Average and standard deviation of the playback quality level of peers over time. 50% of peers fail at time 500 seconds.

5.5.3 Resilience to Catastrophic Failures

Purpose and outcome. We evaluate the impact of the catastrophic failure on the playback quality level of peers remaining in the overlay. Experimental results show that the failure temporarily reduces the average playback quality level by only about 5%, however, after only 3 exploration rounds (i.e., 30 seconds) this average becomes the same as before the failure.

Experimental Setup. Experiments in this section use the same settings as those presented in Section 5.5.1 and additionally simulate the catastrophic failure at time 500 seconds. The failure is simulated in the same way as in Section 5.4.3.

Analysis. Figure 5.24 shows the average and standard deviation of the playback quality level of peers over time. The figure shows that the catastrophic failure temporarily reduces the average playback quality level by about 5% at time 30 seconds after the failure. The little impact of the failure on the playback quality corresponds to the little

impact of the failure on the download rate of peers, as shown in Section 5.4.3. The delay of 30 seconds in the reaction to the failure results from the use of the sliding buffer by peers. A peer downloads the most recent chunks and stores them at the right side of the buffer shown in Figure 4.2. In contrast, leftmost chunks from the buffer are used for playback. When a catastrophic failure occurs, it affects the number of chunks downloaded by a peer at the right side of the peer's buffer. These chunks affect the quality of playback only when the buffer slides such that they become leftmost chunks in the buffer. The time required for such buffer slide corresponds to the length of the buffer, i.e., playback lag time, which is set to 30 seconds in the experiments.

5.5.4 Playback Startup Delay

Purpose and outcome. In this section, we evaluate the playback startup delay at peers joining the system. We show that the playback of basic quality starts at a peer only 3 seconds after the peer joins the system and then the quality of playback gradually improves.

Experimental setup. Experiments in this section use the same settings as those presented in Section 5.5.1 and additionally simulate two peers with different downlink bandwidth, one from category A and one from category D, joining the adapted overlay.

Analysis. Figure 5.25 shows the playback quality level of two peers joining the adapted overlay. When these two peers join the overlay, their buffers are empty and they do not deliver playback. Initially, their $TargetQuality$ value is set to 1, so they download only a single chunk per timestamp. Within only 3 seconds, the two peers download a sufficient amount of chunks for continuous playback at the quality level of 1. This is a very short time compared to existing P2P live streaming systems that may require up to several minutes before the playback can begin [52]. The quality of playback at

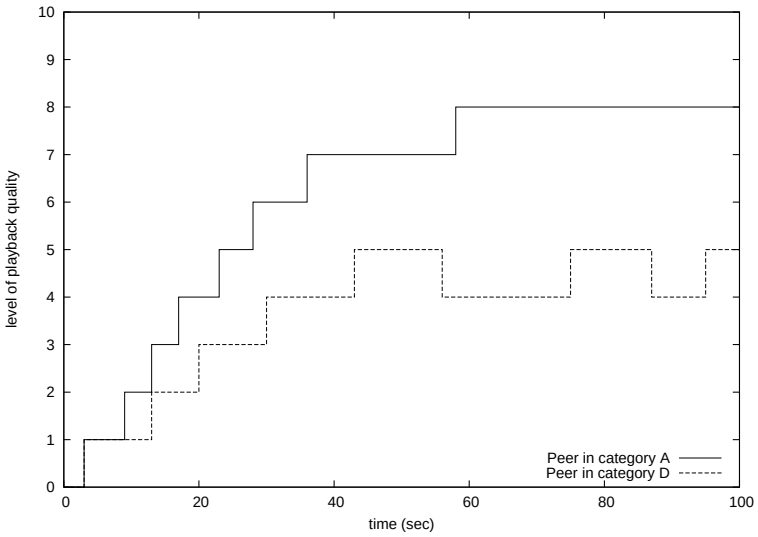


Figure 5.25: Playback quality level of two joining peers.

the two peers gradually increases. The playback quality level of the peer in category A reaches 8 after 60 seconds, whereas the playback quality level of the peer in category D reaches 4 after 30 seconds and then oscillates between 4 and 5. The reason for the small oscillations in the quality of playback is following. The download rate of a peer is typically higher than the one required for playback at the quality level m, but lower than the one required for playback at the quality level $m + 1$, for some integer m between 0 and $M - 1$. Thus, the playback quality level oscillates between m and $m + 1$.

5.6 Discussion

In this chapter, we discussed various approaches for evaluating P2P systems and reasons for the use of a simulator to evaluate MeshTV. We then presented the MeshTV network simulator and the settings for experiments. We showed that MeshTV adapts the overlay so that the out-degree of every peer adapts to the peer's uplink bandwidth, the upload

rate of every peer is maximised and download rates are nearly uniform among peers. We also showed that these download rates are resilient to peer churn and catastrophic failures.

We investigated the delay between the time when the stream is produced at the transmitter and the time when it is delivered to peers. The experimental results show that this delay is nearly uniform among peers and does not exceed 7 seconds for 5000 peers. Then, we evaluated the time required to adapt the mesh overlay. The results show that this time is low and independent of the number of peers in the overlay, thereby ensuring scalability of the overlay adaptation. We analysed the communication overhead of MeshTV and showed that it is about 13% of the media content transmitted in the network.

Furthermore, we limited the downlink bandwidth of peers and showed that the quality of playback at a peer adapts to the peer's download rate. We also showed that the quality of playback is resilient to peer churn and catastrophic failures. Finally, we showed that joining peers exhibit only 3 seconds delay before the start of media playback at the basic quality level.

Chapter 6

Conclusions and Future Work

This thesis presented the MeshTV P2P live media streaming system that is scalable, resilient, delivers playback at the optimal quality, and allows for a short playback start-up delay. To achieve these goals, MeshTV adapts to the dynamic and heterogeneous nature of P2P and Internet environments.

This chapter compares approaches used in the state-of-the-art adaptable P2P live streaming systems to approaches used in MeshTV, summaries the most significant contributions of our work and concludes with a discussion of related research issues that remain open for future work.

6.1 Comparison to the State-of-the-Art

Table 6.1 compares approaches used in the state-of-the-art systems reviewed in Chapter 3 to approaches used in MeshTV. The P2P overlay used in the reviewed systems is based either on multiple multicast trees or a mesh overlay. As discussed in Section 2.3.3, mesh-based overlays are more resilient to peer churn and fluctuations in peer bandwidth compared to tree-based overlays. For this reason, MeshTV uses a mesh-based P2P overlay. In the reviewed systems, the mesh overlay is either directed or

	P2P overlay	Membership management	Content dissemination	Overlay adaptation	Playback quality adaptation
ChunkySpread	multiple multicast trees	random-walk-based	push-based	decentralised algorithm; underutilises bandwidth	no adaptation
Chainsaw	undirected mesh overlay	centralised	pull-based with pipelining	no adaptation	no adaptation
CoolStreaming	undirected mesh overlay	gossip-based	pull-based with periodic scheduling	decentralised algorithm; underutilises bandwidth	no adaptation
PRIME	directed mesh overlay	centralised	pull-based with periodic scheduling	centralised algorithm; utilises the entire uplink bandwidth	unknown algorithm
MeshTV	directed mesh overlay	gossip-based	pull-based with pipelining by peers & push-based by the transmitter	decentralised algorithm; utilises the entire uplink bandwidth	algorithms for peers to adapt the quality of playback to their download rate and to reduce playback startup delay

Table 6.1: Comparison of approaches used in the reviewed systems and in MeshTV.

undirected. A mesh overlay is directed when neighbours of a peer are divided into those that the peer uploads content to and those that the peer downloads content from. In turn, a mesh overlay is undirected when a peer may both upload and download content from any neighbour. PRIME and MeshTV use directed mesh overlays in order to enable peers to change their upload or download rate by changing the number of their uploading or downloading neighbours.

Membership management refers to the problem of how peers discover other peers. Such peer discovery is necessary for a peer to join the system, to replace departing neighbours, or to adapt the P2P overlay. In Chainsaw and PRIME, membership management is realised using a centralised server. However, finite resources of a centralised server may limit the scalability of these systems. Therefore, decentralised membership management, based on random walking and gossipping, is used in the remaining reviewed systems and in MeshTV.

In tree-based systems, such as ChunkySpread, a media (sub)stream is uploaded, i.e., "pushed", to a peer by its parent. Peers download content that they do not actively request, and so this approach to content dissemination is called push-based. In mesh-based systems, a peer downloads chunks from multiple neighbours in parallel. To ensure that neighbours sent different chunks, a peer explicitly selects and requests, i.e., "pulls", chunks from its neighbours. Such approach to content dissemination is called pull-based. While the reviewed mesh-based systems use a pull-based approach, MeshTV uses a combination of push-based and pull-based approaches. A pull-based approach is used by a peer to download chunks from a sender that is not the transmitter. In turn, a push-based approach is used by the transmitter to sent chunks to its direct receivers. The transmitter uses the push-based approach to ensure that it uploads each chunk approximately the same number of times. As discussed in Section 4.2.3, this reduces the stream reception delay at peers.

In the reviewed state-of-the-art systems, two types of pull-based approaches are

used. In CoolStreaming and PRIME, a peer periodically schedules which chunks should be downloaded from which neighbours during the succeeding round. In turn, Chainsaw and MeshTV use a pipelining technique in which a peer distributes requests for chunks across all its neighbours and issues a new request whenever an old one is satisfied. MeshTV uses the pipelining technique because it adapts faster to variations in the bandwidth of peers compared to periodic scheduling.

Overlay adaptation algorithms are used in ChunkySpread, CoolStreaming, PRIME and MeshTV to improve the utilisation of the uplink bandwidth of peers. ChunkySpread uses a decentralised algorithm in which peers adapt the number of their children in multicast trees to their uplink bandwidth. However, to accommodate some oscillations in the uplink bandwidth of a peer without the need to reconstruct multicast trees, ChunkySpread underestimates the uplink bandwidth of peers. This results in some uplink bandwidth remaining unutilised. As discussed in Section 3.4.2 and experimentally shown in Section 5.4, a random mesh overlay does not allow to utilise the entire uplink bandwidth of peers. Therefore, CoolStreaming uses a decentralised algorithm to adapt a mesh overlay so that two peers become neighbours if either of the peers can provide high upload rate to another. However, as discussed in Section 3.5.2, the number of neighbours of peers remains constant and, for this reason, CoolStreaming is unable to use the entire uplink bandwidth of high capacity peers. PRIME and MeshTV propose to adapt the mesh overlay so that the out-degree of peers adapts to their uplink bandwidth. However, the overlay adaptation in PRIME is coordinated by a centralised server with a global knowledge about all participating peers and their bandwidth. Such a centralised approach does not scale to a large and dynamic population of peers. Moreover, the overlay adaptation in PRIME assumes that each peer individually estimates its uplink and downlink bandwidth, whereas such estimation is inaccurate and does not take into account congestion in the core of the Internet. In contrast to PRIME, MeshTV proposes a fully decentralised algorithm for adapting the

mesh overlay. The algorithm is scalable and does not rely on peers estimating their own bandwidth. It adapts the mesh overlay so that download rates are nearly uniform among peers and the upload rate of peers is maximised.

Adaptation of the overlay is required, but not sufficient to deliver playback of the maximum quality at peers. First, the transmitter is not aware of the streaming capacity of the system that depends on the bandwidth available at participating peers. Second, download rates may vary among peers. These variations may be caused by the low downlink bandwidth of some peers reducing their download rate. Small variations may also be caused by peer churn and peer bandwidth fluctuations. An approach to these challenges is to disseminate multiple media descriptions in the overlay and enable peers to adapt the number of downloaded descriptions, and hence the quality of playback, to the download rate of these peers. PRIME proposed this idea, however, it has not proposed an algorithm to accomplish such adaptation. To our knowledge, MeshTV is the first mesh-based system to propose algorithms for such adaptation. In addition to maximising the quality of playback at peers, MeshTV algorithms reduce the playback startup delay at peers joining a transmission. Joining peers initially download a single media description that corresponds to the basic quality of playback and allows for a short startup delay. The quality of playback gradually improves over time as the number of descriptions to download is increased.

6.2 Contributions

The motivation for the work presented in this thesis arose from our observation that state-of-the-art P2P live streaming systems fail to adapt either to heterogeneity or to dynamism of P2P and Internet environments. The heterogeneity comes mainly from differences in the amount of resources available at peers. The dynamism comes mainly from arrival and departure of peers and variations in the amount of resources available

at peers. To accommodate heterogeneity, P2P live streaming systems use overlays based on multiple multicast trees allowing bandwidth of all participating peers to be utilised. To accommodate dynamism, P2P live streaming systems rely on mesh overlays that are highly resilient to peer churn and to varying peer bandwidth. However, multiple multicast trees adapt poorly to dynamism, whereas existing mesh-based approaches underuse the bandwidth of peers.

This thesis began by introducing the main concepts, challenges and methods of streaming media over the Internet. The challenges of media streaming using traditional client-server architectures relate mainly to the large bandwidth requirements of the content provider. These requirements apply to the bigger domain of large-scale content delivery that includes file-sharing, live streaming and on-demand streaming applications. For these applications, P2P approaches enable to significantly reduce the bandwidth requirements of the content provider by using the bandwidth of participating peers. P2P approaches for these applications are introduced in Chapter 2.

Chapter 3 focused on the state-of-the-art approaches to adaptable P2P live media streaming and presented their shortcomings in terms of resilience to peer churn, adaptability to heterogeneous and dynamic bandwidth of peers, and adaptability of playback quality. The first contribution of this work is the finding that existing mesh-based P2P live streaming systems suffer from several inefficiencies. First, a media stream is transmitted at the same quality to all peers. Second, only a portion of the available uplink bandwidth of peers is used for the dissemination of the media stream. Third, download rates are non-uniform among peers. One consequence of these inefficiencies is that some viewers cannot deliver continuous playback because their download rate is below the rate of the media stream, whereas the remaining viewers deliver playback of suboptimal quality. Another consequence is that bandwidth costs of the content provider are not minimised. As most of the state-of-the-art P2P live streaming systems, both academic and commercial, are mesh-based, the findings in this thesis will

have a large impact on their design. It is expected that mechanisms to overcome the discovered inefficiencies will be an integral part of future mesh-based P2P live streaming systems. It is also expected that these findings will initiate further research on the optimality of mesh-based approaches for P2P live streaming. These findings may also motivate equivalent research in the related fields of P2P on-demand streaming and P2P file-sharing, where mesh-based approaches are commonly used.

Chapter 4 described the MeshTV system. MeshTV uses a mesh overlay for its resilience to peer churn and to variations in peer bandwidth. However, in contrast to existing mesh-based approaches, MeshTV adapts the mesh overlay to the heterogeneous bandwidth of peers. The decentralised algorithm for adapting the mesh overlay is the second contribution of this thesis. The algorithm adapts the overlay so that download rates are nearly uniform among peers and the entire uplink bandwidth available at peers is utilised. The significance of this algorithm is that it minimises the usage of the bandwidth of the content provider by maximising the usage of the bandwidth of viewers. As a consequence, it minimises the bandwidth costs of the content provider and allows to maximise the quality of media streams transmitted to viewers. These benefits will be an incentive for designers of P2P live streaming systems to adapt this algorithm into their design. This algorithm will motivate the design of equivalent algorithms for P2P file-sharing and P2P on-demand streaming systems.

We performed a comprehensive simulation analysis of the overlay adaptation algorithm in Section 5.4. The evaluation results show that the algorithm adapts the mesh overlay in a short time that is independent of the number of peers in the overlay. For example, for the bandwidth distribution similar to that in the real-world and the peer in-degree of 10, an initially random overlay adapts in only 4 adaptation rounds. The results also show that the overlay adaptation is resilient to peer churn and to failure of a large number of peers. Moreover, in the adapted overlay with 5000 peers, the media stream is delivered to each peer with approximately the same delay of about 7 seconds.

Finally, the algorithm requires little communication overhead.

Another contribution of this thesis are the algorithms for peers to adapt the quality of their playback to their download rate. These algorithms improve the experience of viewers by maximising the individual quality of their playback and reducing their playback startup delay. We expect that these benefits will motivate designers of P2P live streaming systems to incorporate the algorithms in this thesis into their design. The algorithms of this thesis may also be applicable to P2P on-demand streaming systems, which would benefit from maximised individual quality of playback of each viewer and reduced playback startup delay.

We performed a comprehensive simulation analysis of the playback quality adaptation algorithms in Section 5.5. The evaluation results show that the quality of playback at peers is resilient to peer churn and to failures of large numbers of peers. The results also show that the algorithms reduce the startup delay to about 3 seconds for the playback at the basic quality.

A further contribution of this thesis is the packet-level network simulator developed for the analysis of MeshTV in Chapter 5. The simulator has been developed due to the lack of a network simulator capable of accurately simulating high-bandwidth data transmission between a large number of nodes. The network simulator is useful for simulating all large-scale distributed systems. In particular, it is useful for simulating P2P content delivery systems, such as P2P file-sharing, P2P live streaming and P2P on-demand streaming systems. Therefore, we plan to release it to the public.

6.3 Open Research Issues

As is the case with research, there are some challenges that remain for possible future work. One such challenge is to adapt a mesh overlay so that the delay in the reception of a media stream by peers is minimised. In this thesis, we investigated this delay

in MeshTV and showed that it is low and nearly uniform among peers. In [13], we proposed an algorithm that adapts a mesh overlay so that this delay is further reduced. However, this algorithm is incompatible with the overlay adaptation algorithm that is used in MeshTV to adapt the out-degree of peers to their uplink bandwidth. Incompatibility of the two algorithms results from conflicting modifications performed by these algorithms on the overlay. A design and evaluation of a single overlay adaptation algorithm that integrates the two algorithms remains open for future work.

Furthermore, the increasing popularity of P2P systems has become a concern for ISPs. P2P live media streaming systems tend to increase the traffic of ISPs, as peers first download content and then upload the content to other peers. This generates approximately double the amount of traffic at the ISPs compared to client-server architectures, where most of the traffic flows in one direction. As a consequence, network traffic costs of ISPs may increase. This motivates research in locality-aware P2P systems, such as [40, 136, 70, 19], that exploit network proximity between peers to mitigate the impact of P2P on ISPs. Adaptation of the MeshTV overlay to provide locality-awareness remains open for future work.

Challenging research problems in the area of P2P live streaming are secure streaming and incentives. Secure streaming relates to preventing peers from disrupting the correct behaviour of the system. This may involve malicious peers altering the media stream, preventing other peers from receiving the media stream, or partitioning the P2P overlay. In turn, incentives relate to preventing peers from acting selfishly for their individual benefit. Peers may refuse to forward media streams to other peers in order to save their own uplink bandwidth or peers may attempt to take advantage of the system by downloading at a higher rate compared to other participating peers. This type of non-cooperative behaviour may result in the "tragedy of the commons" [49], when the correct functioning of the system becomes impossible. To address this, systems may use incentives for peers to cooperate. Incentives may be designed to en-

sure that peers upload approximately the same amount of content as they download [22, 83, 122, 80]. However, such strict fairness may not be desirable in P2P live streaming systems as it prevents the possibility that some peers are willing to contribute more bandwidth than they consume. It also rejects peers that cannot share fairly due to network configurations and firewalls. Therefore, different types of incentives have been proposed for P2P live streaming. For instance, [90] proposes to reward peers that contribute more uplink bandwidth with shorter delays in the reception of the media stream. In turn, [74] proposes to reward peers that contribute more uplink bandwidth with a higher quality of their media playback. Extending MeshTV to incorporate secure streaming and incentives remains open for future work.

Bibliography

[1] A. Adams, J. Nicholas, and W. Siadak. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification. RFC 3973, Jan. 2005.

[2] B. Adamson, C. Bormann, M. Handley, and J. Macker. Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol. RFC 3940, Nov. 2004.

[3] Akamai website. `http://www.akamai.com`, [Accessed 10 Sep. 2008].

[4] Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper. IANA Guidelines for IPv4 Multicast Address Assignments. RFC 3171, Aug. 2001.

[5] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[6] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. R. Rodriguez. Is high-quality VoD feasible using P2P swarming? In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 903–912. ACM Press, 2007.

[7] R. Aravind, M. R. Civanlar, and A. R. Reibman. Packet loss resilience of MPEG2 scalable video coding algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(5):426–435, Oct. 1996.

[8] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long. Managing flash crowds on the Internet. In *Proceedings of the 11th Conference on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*, pages 246–249. IEEE Press, 2003.

[9] S. Baset and H. Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 1–11, 2006.

[10] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving a BitTorrent network's performance mechanisms. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, 2006.

[11] B. Biskupski, R. Cunningham, J. Dowling, and R. Meier. High-bandwidth mesh-based overlay multicast in heterogeneous environments. In *AAA-IDEA '06: Proceedings of the 2nd International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*, pages 4–11. ACM Press, 2006.

[12] B. Biskupski, R. Cunningham, and R. Meier. Improving throughput and node proximity of P2P live video streaming through overlay adaptation. In *Proceedings of the 9th IEEE International Symposium on Multimedia (ISM 2007)*, pages 245–252. IEEE Computer Society, 2007.

[13] B. Biskupski, M. Schiely, P. Felber, and R. Meier. Tree-based analysis of mesh overlays for peer-to-peer streaming. In *Proceedings of the 8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2008)*, pages 126–139, 2008.

[14] L. Bracciale, F. L. Piccolo, S. Salsano, and D. Luzzi. Simulation of peer-to-peer streaming over large-scale networks using opss. In *ValueTools '07: Proceedings of*

the 2nd international conference on Performance evaluation methodologies and tools, pages 1–10. ICST, 2007.

[15] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376, Oct. 2002.

[16] C. Caini and R. Firrincieli. TCP Hybla: A TCP enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking*, 22(5):547–566, Aug. 2004.

[17] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth multicast in cooperative environments. In *SOSP'03: Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 298–313, 2003.

[18] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized publish-subscribe infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1489–1499, Oct. 2002.

[19] Y. Chen, B. Deng, and X. Li. Rainbow: A locality-aware peer-to-peer overlay multicast system. In *GCCW '06: Proceedings of the 5th International Conference on Grid and Cooperative Computing Workshops*, pages 151–155. IEEE Computer Society, 2006.

[20] Y. H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast*, 20(8):1456–1471, Oct. 2002.

[21] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the International Workshop on Designing Privacy Enhancing Technologies*, pages 46–66. Springer-Verlag, 2000.

[22] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[23] Y. Cui, B. Li, and K. Nahrstedt. oStream: Asynchronous streaming multicast in application-layer overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1):91–106, 2004.

[24] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 202–215, Oct. 2001.

[25] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems (TOCS)*, 8(2):85–110, 1990.

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

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

[28] P. Druschel and A. I. T. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, pages 75–80. IEEE Computer Society, 2001.

[29] Feidian website. `http://www.feidian.com`, [Accessed 10 Sep. 2008].

[30] P. Felber and E. W. Biersack. Cooperative content distribution: Scalability through self-organization. In *Self-star Properties in Complex Information Systems*, pages 343–357. Springer-Verlag, 2005.

[31] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, Jan. 1968.

[32] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification. RFC 4601, Aug. 2006.

[33] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[34] Folding@home distributed computing. `http://folding.stanford.edu`, [Accessed 10 Sep. 2008].

[35] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *ATEC '05: Proceedings of the USENIX Annual Technical Conference*, pages 13–13. USENIX Association, 2005.

[36] C. P. Fry and M. K. Reiter. Really truly trackerless BitTorrent. Technical Report CMU-CS-06-148, School of Computer Science, Carnegie Mellon University, 2006.

[37] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, 2003.

[38] P. Garbacki, D. H. J. Epema, J. Pouwelse, and M. van Steen. Offloading servers with collaborative video on demand. In *Proceedings of the 7th International Workshop on Peer-to-Peer Systems (IPTPS'08)*, Feb. 2008.

[39] P. Garbacki, A. Iosup, D. H. J. Epema, and M. van Steen. 2Fast: Collaborative downloads in P2P networks. In *Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P2006)*, Sept. 2006.

[40] L. Garces-Erice and E. W. Biersack. MULTI+: A robust and topology-aware peer-to-peer multicast service. *Computer Communications*, 29(7):900–910, Apr. 2006.

[41] C. GauthierDickey and C. Grothoff. Bootstrapping of peer-to-peer networks. In *Proceedings of the International Workshop on Dependable and Sustainable Peer-to-Peer Systems*. IEEE, August 2008.

[42] M. Ghanbari. Two-layer coding of video signals for VBR networks. *IEEE Journal on Selected Areas in Communications*, 7:801–806, June 1989.

[43] T. J. Giuli and M. Baker. Narses: A scalable flow-based network simulator. Technical report, Stanford University, 2002.

[44] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 2235–2245. IEEE, 2005.

[45] V. K. Goyal. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, 18(5):74–93, Sept. 2001.

[46] Y. Gu and R. L. Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(7):1777–1799, 2007.

[47] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2Cast: Peer-to-peer patching scheme for VoD service. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 301–309. ACM Press, 2003.

[48] Y. Guo, K. Suh, J. Kurose, and D. Towsley. DirectStream: A directory-based peer-to-peer video streaming service. *Computer Communications*, 31(3):520–536, 2008.

[49] G. Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243–1248, 1968.

[50] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. Insights into PPLive: A measurement study of a large-scale P2P IPTV system. In *Proceedings of IPTV Workshop, International World Wide Web Conference*, 2006.

[51] X. Hei, Y. Liu, and K. W. Ross. Inferring network-wide quality in P2P live streaming systems. *IEEE Journal on Selected Areas in Communications*, 25(9):1640–1654, 2007.

[52] X. Hei, Y. Liu, and K. W. Ross. IPTV over P2P streaming networks: The mesh-pull approach. *IEEE Communications Magazine*, 46(2):86–92, 2008.

[53] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. In *Proceedings of the ACM International Conference on Multimedia*, pages 191–200, 1998.

[54] Ipoque. Internet study 2007. `http://www.ipoque.com/resources/internet_studies/internet_study_2007`, Oct. 2007 [Accessed 10 Sept. 2008].

[55] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *Proceedings of the 5th International Workshop on Passive and Active Measurements*, Apr. 2004.

[56] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 25(1):157–187, 1995.

[57] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole. Overcast: reliable multicasting with an overlay network. In *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI'00)*. USENIX Association, 2000.

[58] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, pages 79–98. Springer-Verlag New York, Inc., 2004.

[59] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Dept. of Computer Science, Vrije Universiteit Amsterdam, The Netherlands, 2003.

[60] Joost website. `http://www.joost.com`, [Accessed 10 Sep. 2008].

[61] D. Kostic, R. Braud, C. Killian, E. Vandekieft, and J. W. Anderson. Maintaining high bandwidth under dynamic network conditions. In *Proceedings of 2005 USENIX Annual Technical Conference*, pages 193–208, 2005.

[62] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 282–297, 2003.

[63] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, Nov. 2000.

[64] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for P2P streaming systems. In *Proceedings of the 26th IEEE International Conference on Computer Communication (INFOCOM)*, pages 919–927, May 2007.

[65] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoğlu, D. Estrin, and M. Handley. The MASC/BGMP architecture for inter-domain multicast routing. *ACM SIGCOMM Computer Communication Review*, 28(4):93–104, 1998.

[66] K. Lai and M. Baker. Measuring bandwidth. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 235–245, 1999.

[67] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, 1997.

[68] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the Kazaa network. In *Proceedings of the 3rd Workshop on Internet Applications*, pages 112–120. IEEE Computer Society, June 2003.

[69] J. Li. On peer-to-peer (P2P) content delivery. *Peer-to-Peer Networking and Applications*, 1(1):45–63, Mar. 2008.

[70] J. Liang and K. Nahrstedt. DagStream: Locality aware and failure resilient peer-to-peer streaming. In *Proceedings of the 13th SPIE/ACM Conference on Multimedia Computing and Networking*. SPIE/ACM, 2006.

[71] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *PODC '02: Proceedings of the 21th Annual Symposium on Principles of Distributed Computing*, pages 233–242. ACM Press, 2002.

[72] J. Liu, S. G. Rao, B. Li, and H. Zhang. Opportunities and challenges of peer-to-peer Internet video broadcast. *Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 96(1):11–24, 2008.

[73] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, Mar. 2008.

[74] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang. P2P video live streaming with MDC: Providing incentives for redistribution. In *Proceedings of the 2007 IEEE International Conference on Multimedia and Expo, ICME 2007*, pages 48–51, 2007.

[75] N. Magharei and R. Rejaie. PRIME: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 1415–1423, 2007.

[76] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 1424–1432, 2007.

[77] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, and M. Roccetti. TCP Libra: Exploring RTT-fairness for TCP. In *Proceedings of the IFIP Conference on Networking*, volume 4479, pages 1005–1013. Springer, 2007.

[78] L. Massoulié and M. Vojnović. Coupon replication systems. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, volume 33, pages 2–13. ACM Press, June 2005.

[79] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the International Workshop on Peer-To-Peer Systems (IPTPS)*, pages 53–65. Springer-Verlag, 2002.

[80] J.-D. Mol, D. H. J. Epema, and H. J. Sips. The Orchard algorithm: P2P multicasting without free-riding. In *Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing*, pages 275–282, 2006.

[81] A. Montresor. A robust protocol for building superpeer overlay topologies. In *Proceedings of the Conference on Peer-to-Peer Computing*, pages 202–209, 2004.

[82] J. Moy. Multicast Extensions to OSPF. RFC 1584, Mar. 1994.

[83] A. Nandi, T.-W. Ngan, A. Singh, P. Druschel, and D. S. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. In *Proceedings of the ACM/IFIP/USENIX International Middleware Conference*, pages 270–291, 2005.

[84] The network simulator - ns-2. `http://www.isi.edu/nsnam/ns`, [Accessed 10 Sep. 2008].

[85] A. Odlyzko. Data networks are lightly utilized, and will stay that way. *Review of Network Economics*, 2(3):210–237, Sept. 2003.

[86] T. Oelbaum, V. Baroncini, T. K. Tan, and C. Fenimore. Subjective quality assessment of the emerging AVC/H.264 video coding standard. In *Proceedings of the International Broadcasting Conference (IBC)*, Sept. 2004.

[87] Omnet++ website. `http://www.omnetpp.org`, [Accessed 10 Sep. 2008].

[88] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient peer-to-peer streaming. In *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols*, pages 16–27. IEEE Computer Society, 2003.

[89] V. S. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of the International Workshop on Peer-To-Peer Systems (IPTPS)*, pages 127–140, 2005.

[90] F. Pianese, D. Perino, J. Keller, and E. W. Biersack. PULSE: An adaptive, incentive-based, unstructured P2P live streaming system. *IEEE Transactions on Multimedia*, 9(8):1645–1660, 2007.

[91] PlanetLab website. `http://www.planet-lab.org`, [Accessed 10 Sep. 2008].

[92] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P file-sharing system: Measurements and analysis. In *Proceedings of the 4th International Workshop on Peer-To-Peer Systems (IPTPS'05)*, Feb. 2005.

[93] PPLive website. `http://www.pplive.com`, [Accessed 10 Sep. 2008].

[94] PPStream website. `http://www.ppstream.com`, [Accessed 10 Sep. 2008].

[95] R. Puri and K. Ramchandran. Multiple description source coding through forward error correction codes. In *Proceedings of the 33rd Asilomar Conference on Signals, Systems, and Computers*, volume 1, pages 342–346. IEEE Computer Society, Jan. 1999.

[96] R. Puri, K. Ramchandran, K. Lee, and V. Bharghavan. Forward error correction (FEC) codes based multiple description coding for internet video streaming and multicast. *Signal Processing: Image Communication*, 16(8):745–762, May 2001.

[97] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 367–378. ACM Press, 2004.

[98] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 161–172. ACM Press, 2001.

[99] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proceedings of the 2004 USENIX Annual Technical Conference*, pages 127–140. USENIX, June 2004.

[100] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[101] J. Ritter. Why Gnutella Can't Scale. No, Really. `http://www.darkridge.com/~jpr5/doc/gnutella.html`, Feb. 2001, [Accessed 10 Sept. 2008].

[102] P. Rodriguez and E. W. Biersack. Dynamic parallel access to replicated content in the Internet. *IEEE/ACM Transactions on Networking*, 10(4):455–465, 2002.

[103] P. Rodriguez, S.-M. Tan, and C. Gkantsidis. On the feasibility of commercial, legal P2P content distribution. *ACM SIGCOMM Computer Communication Review*, 36(1):75–78, 2006.

[104] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350. Springer-Verlag, 2001.

[105] S. Saroiu, K. P. Gummadi, and S. D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *ACM Multimedia Systems*, 9(2):170–184, 2003.

[106] M. Schiely and P. Felber. Peer-to-peer distribution architectures providing uniform download rates. In *Proceedings of the International Symposium on Distributed Objects and Applications (DOA'05)*, pages 1083–1096, 2005.

[107] M. Schiely, L. Renfer, and P. Felber. Self-organization in cooperative content distribution networks. In *NCA '05: Proceedings of the 4th IEEE International Symposium on Network Computing and Applications*, pages 109–118. IEEE Computer Society, 2005.

[108] SETI. SETI@home: Search for extraterrestrial intelligence at home. `http://setiathome.ssl.berkeley.edu`, [Accessed 10 Sept. 2008].

[109] A. Sharma, A. Bestavros, and I. Matta. dPAM: A distributed prefetching protocol for scalable asynchronous multicast in P2P systems. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 1139–1150. IEEE, 2005.

[110] M. Shirts and V. S. Pande. Screen savers of the world unite! *Science*, 290:1903–1904, 2000.

[111] SopCast website. `http://www.sopcast.com`, [Accessed 10 Sep. 2008].

[112] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano. PGM Reliable Transport Protocol Specification. RFC 3208, Dec. 2001.

[113] N. Spring, L. Peterson, A. Bavier, and V. Pai. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Operating Systems Review*, 40(1):17–24, 2006.

[114] SSFNet. Scalable Simulation Framework Net. `http://www.ssfnet.org`, [Accessed 10 Sept. 2008].

[115] W. R. Stevens. *TCP/IP illustrated (vol. 1): the protocols.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[116] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[117] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *IMC'03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 39–44, 2003.

[118] D. A. Tran, K. A. Hua, and T. T. Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, volume 2, pages 1283–1292, 2003.

[119] TVants website. `http://www.tvants.com`, [Accessed 10 Sep. 2008].

[120] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. *SIGOPS Operating Systems Review*, 36(SI):271–284, 2002.

[121] V. Venkataraman, K. Yoshida, and P. Francis. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 2–11. IEEE Computer Society, 2006.

[122] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A secure economic framework for P2P resource sharing. In *Proceedings of the 1st Workshop on the Economics of Peer-to-Peer Systems*, June 2003.

[123] V. Vishnumurthy and P. Francis. On heterogeneous overlay construction and random node selection in unstructured P2P networks. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, 2006.

[124] A. Vitali and M. Fumagalli. Standard-compatible multiple-description coding (MDC) and layered coding (LC) of audio/video streams. Internet Draft - Network Working Group, July 2006.

[125] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiToS: Enhancing BitTorrent for supporting streaming applications. In *Proceedings of the 9th IEEE Global Internet Symposium*, Apr. 2006.

[126] S. Voulgaris, D. Gavidia, and M. van Steen. CYCLON: Inexpensive membership management for unstructured P2P overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005.

[127] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. RFC 1075, Nov. 1988.

[128] D. Werthimer, J. Cobb, M. Lebofsky, D. Anderson, and E. Korpela. SETI@home-massively distributed computing for SETI. *Computing in Science and Engineering*, 3:78–83, Jan. 2001.

[129] A. Whitaker and D. Wetherall. Forwarding without loops in Icarus. In *Proceedings of the IEEE Conference on Open Architectures and Network Programming*, pages 63–75, 2002.

[130] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, pages 255–270. USENIX Association, Dec. 2002.

[131] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering*, pages 49–60. IEEE Xplore, Mar. 2003.

[132] W. Yang and N. Abu-Ghazaleh. GPS: A general peer-to-peer simulator and its use for modeling BitTorrent. In *MASCOTS '05: Proceedings of the 13th IEEE*

International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pages 425–434. IEEE Computer Society, 2005.

[133] C. K. Yeo, B. S. Lee, and M. H. Er. A survey of application level multicast techniques. *Computer Communications*, 27(15):1547–1568, Sept. 2004.

[134] Zattoo website. `http://www.zattoo.com`, [Accessed 10 Sep. 2008].

[135] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. CoolStreaming/DONet: A data-driven overlay network for live media streaming. In *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, pages 2102–2011, 2005.

[136] X. Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu. A construction of locality-aware overlay network: mOverlay and its performance. *IEEE Journal on Selected Areas in Communications*, 22(1):18–28, Jan. 2004.

[137] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, Jan. 2004.