Live Streaming Performance of Peer-to-Peer Systems

ILIAS CHATZIDROSSOS

KTH Electrical Engineering

Doctoral Thesis
Stockholm, Sweden 2012

KTH Electrical Engineering

Live Streaming Performance of Peer-to-Peer Systems

ILIAS CHATZIDROSSOS

Doctoral Thesis
Stockholm, Sweden, 2012

TRITA-EE 2012:004
ISSN 1653-5146
ISBN 978-91-7501-241-4

School of Electrical Engineering
KTH, Stockholm, Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av doktorsexamen torsdagen den 9 februari 2012 i Sal F3, KTH, Stockholm.

Tryck: Universitetsservice US AB

Abstract

In peer-to-peer (P2P) live streaming systems peers organize themselves in an overlay and contribute with their resources to help diffuse live content to all peers in a timely manner. The performance of such systems is usually characterized by the delay-loss curve, which quantifies the playback delay required for achieving a certain streaming quality, expressed as the chunk missing ratio at the peers. The streaming quality is determined by the overlay construction algorithm, the forwarding algorithm, the loss process in the underlying network, the number of peers in the overlay and their bandwidth distribution, the willingness of the peers to contribute with their resources and the viewing behavior of the peers (churn). The overlay construction and forwarding algorithms are inherent characteristics of a P2P protocol, while the remaining factors are artifacts of the deployment of the P2P system over a best-effort network such as the Internet, as well as the fact that peers act as independent agents. The current thesis addresses the problem of evaluating and improving the performance of P2P streaming protocols based on models of the network and of the peers' behavior.

The first part of the thesis is devoted to the performance evaluation of P2P overlay construction and forwarding algorithms and offers three contributions. First, we study the efficiency of data distribution in multiple tree-based overlays employing forward error correction. We derive analytical expressions for the average packet possession probability as well as its asymptotic bounds and verify our results through simulations. Second, we evaluate the performance of a streaming system in the presence of free-riders. We define two admission control policies and study the streaming feasibility using an analytical model and via simulations. Third, we present an analytic framework for the evaluation of forwarding algorithms in mesh-based systems. We validate it via simulations and use it to evaluate and to compare four push-based forwarding algorithms in terms of their delay-loss curves.

The second part of the thesis investigates potential improvements to the operation of P2P streaming systems and offers three contributions in that area. First, we study the impact of selfish peer behavior on streaming quality in overlays where a fraction of peers has limited contribution due to physical constraints. We show that selfish peer behavior results in suboptimal streaming quality and we propose an incentive mechanism that increases the streaming quality by using the server upload capacity to reward high contributing peers. Second, we study the problem of building network aware P2P streaming overlays, taking into account recent measurement results that indicate that the AS-level topology of the Internet is flattening. Through extensive simulations on regular and measured topologies we show that it is possible to create better than random overlays relying on information about the underlying topology. Finally, we study the problem of playout adaptation in P2P streaming systems under churn. We propose and evaluate two algorithms that tune the playback delay of the peers in such a way that the streaming quality of the peers is maintained within predetermined limits. We use simulations to show the correctness of the proposed algorithms and the benefits from their use.

Acknowledgements

I would like to thank my main advisor Ass. Prof. György Dán for his guidance and all the interesting discussions that were providing me with invaluable insights into the problems that I had to address. I highly value his understanding, patience and persistence when things would not go as expected. I would also like to thank my now second, but originally main advisor, Assoc. Prof. Viktória Fodor, for introducing me into the world of research and guiding me through the important first steps of a long journey leading to the writing of this thesis. A big part of the work presented herein was conducted under her supervision. Furthermore, I am thankful to Prof. Gunnar Karlsson for giving me the opportunity to become a member of this lab. Moreover, I would like to thank Dr. Arnaud Legout, my supervisor during my internship at INRIA and co-author of one of the papers included in this thesis. I feel that I should also thank all the members of LCN, current and past, for maintaining a friendly environment in the lab and for breaking the monotony of everyday work.

I am thankful to all my friends everywhere. To those in Stockholm for making me feel at home away from home. To those back home for always being there for me when needed and for constantly showing me that I am still part of their lives despite my six years long physical absence. To those scattered abroad for the same reasons, as well as for hosting me during fun and relaxing/refreshing weekends. Lastly, I would like to express my gratefulness to my parents for their constant support.

Contents

Contents **vii**

1 Introduction **1**

2 Performance metrics and challenges **5**
2.1 Performance metrics 5
2.2 Design challenges for P2P streaming systems 7

3 Architectures **9**
3.1 Tree-based streaming 9
3.2 Mesh-based streaming 13
3.3 Hybrid systems 15

4 Performance evaluation **17**
4.1 Data forwarding in P2P streaming systems 17
4.2 Streaming in heterogeneous P2P networks 20
4.3 Playout adaptation in P2P systems 23
4.4 Network awareness 24

5 Summary of original work **27**

6 Conclusion and Future Work **33**

Bibliography **37**

Paper A: Streaming Performance in Multiple-tree-based Overlays
Paper B: Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates
Paper C: On the Effect of Free-riders in P2P Streaming Systems
Paper D: Server Guaranteed Cap: An incentive mechanism for maximizing streaming quality in heterogeneous overlays
Paper E: Playout adaptation for P2P streaming systems under churn
Paper F: Small-world Streaming: Network-aware Streaming Overlay Construction Policies for a Flat Internet

Chapter 1

Introduction

The introduction of new and efficient multimedia encoding standards combined with the transition to the broadband Internet has led to a constant increase in the demand for multimedia services, notably video, over the past decade. One of the emerging applications has been the transmission of live video content from a source to many destinations following the paradigm of scheduled TV broadcasting. This type of transmission, referred to as multicast, primarily aims to provide TV services over the Internet. It is however not only restricted to that, as it could be used in other contexts as well such as teleconferencing (e.g. Skype).

The most straightforward approach to video delivery from a source to many destinations is based on the client-server paradigm. A user interested in the content connects to the server and then the server streams the video. This approach has two major drawbacks. First, it does not scale well in the number of viewers. The bandwidth requirements of the server increase linearly in the number of viewers and addressing larger audiences requires expensive investments in server infrastructure. Second, it incurs an unecessary stress on the physical network, since the content is transmitted over the access link of the server a number of times equal to the number of viewers.

Efficient one-to-many communication with respect to network resources and scalability can be achieved through the use of IP multicast [1]. In the IP multicast paradigm, users subscribe to multicast groups and the delivery of the content from a source to the group members is done by forming a delivery tree on the network layer, where routers are internal nodes and members of the multicast group are leaf nodes of the tree. In IP multicast, data traverse a physical link only once leading thus to the most efficient usage of network resources. It requires though the deployment of IP multicast-enabled routers that maintain group membership information and perform the routing of data to the members of the multicast group. Although IP multicast is nowadays used for the delivery of IPTV services within the boundaries of many Internet Service Providers (ISP), the lack of deployment of interdomain multicast routing protocols [2] inhibited its prevalence as a mechanism

of delivering live content to users on a global scale.

The failure of IP multicast to become the universal solution turned the interest to solutions run on the application layer of the Internet protocol stack and led to the deployment of Content Delivery Networks (CDN) [3, 4]. CDNs mimic the IP-multicast functionality, but on the application layer and consist of a number of *core* and *replica* servers. Replica servers are strategically positioned servers that the end-users can connect to and receive the streaming content. Core servers are responsible for forwarding the content from the content source to the replica servers, offering application layer routing services. Core servers do not stream the content to end-users. CDNs achieve a lower redundancy on the physical network compared to the client-server delivery, but only partially solve the problem of scalability. An increase in the demand for streaming services still requires the further deployment of servers, which is both costly and time consuming.

Peer-to-peer (P2P) systems were proposed as a potential solution to the scalability problems of IP multicast and CDNs. The success of P2P systems for file sharing paved the way for the introduction of the P2P paradigm for streaming live content. In P2P live streaming systems peers (users-viewers) organize themselves in an overlay and use their resources, bandwidth and processing power, to help data distribution. In short, peers download the live content from some peers and at the same time they upload it to other peers. The scalability of such system is potentially unlimited. The conception of P2P streaming is fairly simple and intuitive. Furthermore, the already acquired experience from file sharing has led to the quick development of a plethora of P2P streaming systems. In fact, P2P streaming has been one of the areas where implementation preceded, to a large extent, mathematical modelling and analysis of the systems.

The ease of deployment and the high global demand for streaming services [5] has helped boost the popularity of P2P streaming systems. It revealed however also many differences in the operation and performance of these systems [6, 7, 8]. These findings raise the question of whether it is possible to build optimal P2P streaming systems and more importantly, if yes, how to build them. Answering this question requires a deep understanding of the fundamental properties of P2P streaming systems and how these are affected by the environment in which the system is deployed. This investigation is challenging in itself for three reasons. The first reason is the large state space for the P2P system design. The second reason is the stochasticity introduced to the P2P system by the Internet as well as by the fact that peers act as independent agents. The third reason is the fact that the interactions and inter-dependencies among the peers make analytical tractability difficult to achieve.

This thesis presents a study of the performance of P2P live streaming systems using analytical methods as well as extensive simulations. Using these tools we aim to:

- Analyze the efficiency of live content delivery in P2P streaming systems.

- Investigate the impact of the resource heterogeneity of the peers on the system performance and to propose solutions to leverage peer contribution.
- Optimize the video delivery efficiency by taking into account information about the underlying network.

The thesis is structured as follows. In Chapter 2, we introduce the most important performance metrics of P2P systems and present the challenges that the P2P system designer has to face. In Chapter 3, we present the P2P streaming system architectures that are used in this thesis. In Chapter 4, we discuss the contributions of this thesis. Chapter 5 contains a summary of the papers included in this thesis and the detailed contribution of the author of the thesis to each of them. Chapter 6 summarizes the main findings and conclusions drawn and discusses potential directions for future research in the area of P2P streaming.

Chapter 2

Performance metrics and challenges

2.1 Performance metrics

The performance of a P2P streaming system can be viewed from two different perspectives: the peer and the system perspective. From the peer point of view the performance of the system is measured by the quality of the delivered video stream. From the system's perspective the performance of the P2P system is measured by the efficiency of the P2P protocol in utilizing the available resources, the fairness in the resource allocation and the impact on the underlying network.

Peer-level performance metrics

The performance of a P2P system in live content distribution is captured by the efficiency in delivering live content to the peers with no errors and at low delay. Some of the most common metrics that we use in this thesis are the following:

Playout/Chunk miss ratio is the ratio of data not received by the time it should be decoded and played out at the peer over the total amount of video data sent by the streaming server. Although the playout miss ratio is the most widely used metric for streaming performance, at least within the P2P streaming community, it is rather generic and does not capture the impact of losses on the perceived video quality. Metrics that are specifically targeted to measure video distortion are the Mean Square Error (MSE) and the Peak Signal-to-Noise Ratio (PSNR). Both of them are calculated from the received video sequence using the initial video sequence as a reference. MSE is the average of the square of the difference of the received and the initial frames of the video taken pixel-by-pixel, while the PSNR is defined as $PSNR = 10log_{10}(\frac{p_{max}^2}{MSE})$ where p_{max} is the maximum possible pixel value of a frame.

Startup delay is the time it takes from the moment a user requests a stream until

the stream starts to be played out on her screen. Startup delay is determined by two factors: channel setup time and buffering delay. During channel setup, a peer looks for peers that are watching the same stream and that can serve as suppliers of the stream. After channel setup a peer does not start to play out the received content immediately, but initially buffers the content. Buffering is required in order to absorb variations in packet delivery times and to increase the probability of seamless playback. A metric related to the startup delay, is the *zapping delay*, which is the time it takes from the moment a user switches channel until the new channel projects onto her screen.

Playback delay is the time difference between the time instant that a video chunk is generated at the server and the time instant this chunk is played out at a peer. The lower the playback delay is, the closer to realtime is the viewing experience of the users.

Playback lag is the time difference between the playing out of the same chunk in two different peers in the overlay. Playback lag is not an issue in the case of streaming pre-recorded video, but it could be a significant source of dissatisfaction when live streaming is considered.

System-level performance metrics

The system-level performance metrics aim at capturing the efficiency of the utilization of resources (mainly upload bandwidth) in the overlay and how well the overlay adapts to the underlying network. Some metrics that we use in the thesis are the following:

Upload Bandwidth Utilization is the ratio of the upload bandwidth of the participating peers that is used by the system over the total available upload bandwidth. A well designed P2P streaming system should be able to use all the upload bandwidth, if necessary, to satisfy the download rate requirements of the peers in the overlay.

Bandwidth Usage Efficiency shows how close a system performs to the optimal system for a given upload bandwidth distribution. An optimal system could be defined, for example, as the one that maximizes the sum of the utilities of the participating peers for a given upload bandwidth distribution.

Network Awareness is a measure of the extent to which the P2P system takes into account information about the underlying network during overlay construction and chunk forwarding. Network awareness is usually viewed with respect to capacity and locality. In capacity aware P2P systems overlay construction and/or data distribution are modified according to the information on the capacities of the participating peers, e.g., prioritization of high capacity peers during video forwarding. Locality awareness can be examined from two different perspectives. First, it can be defined as the amount of inter-AS traffic generated by the P2P system, hence the cost incurred to the AS managing entities. Second, it can be viewed as the stress imposed on the physical network by the P2P overlay, that is the amount of

bandwidth and the length of the network paths used to carry the overlay traffic.

2.2 Design challenges for P2P streaming systems

Ideally, a P2P streaming system should achieve good performance with respect to all of the aforementioned metrics. There are, however, a number of challenges that have to be dealt with and which are related to the environment in which the P2P system operates and to the user behavior.

First, a P2P system has to be *resilient to churn*, which is the joining and leaving of peers in the overlay. Churn can affect the playback continuity of the stream since peer departures may slow down the distribution of data in the overlay. Therefore, the overlay maintenance protocol has to locate new suppliers as fast as possible to alleviate the impact of a departure on the playback continuity of a peer.

Second, a P2P streaming system should scale well with the number of participating peers. As the overlay increases in size, the delay of delivering video data to all peers increases. For maintaining a constant playout miss ratio as the overlay increases in size, the playback delay of the peers has to increase as well. As far as scalability is concerned the challenge is two-fold. First, the increase in the average playback delay as the overlay size increases should be such that it does not have a severe impact on the streaming experience of the peers. Second, the system should maintain a small playback lag among peers, so as not to affect the live experience.

The third challenge of a P2P streaming system design is the utilization of heterogeneous bandwidth resources. Streaming feasibility depends on the so-called resource index, which is the ratio of the average upload bandwidth contribution over the video encoding rate. If the resource index is larger or equal to 1, then streaming at full rate to all peers is theoretically feasible. In todays Internet the upload bandwidths of the peers can be very different with respect to the streaming rate and, moreover, the asymmetric nature of some access technologies can lead to low resource index values. Therefore, P2P systems need to build overlays where the heterogeneous bandwidth resources are effectively used.

Fourth, apart from physical limitations, the resource index can also be constrained by the unwillingness of peers to contribute. Peers that want to receive data but not participate in the video dissemination are called free-riders. Free-riders are a threat to the system because they consume resources without contributing any, thus lowering the resource index. Therefore, it is important to design P2P systems that provide incentives to contribute.

Fifth, a P2P system needs to maximize the social welfare in the overlay given by the sum of the peers' utility functions. This is particularly important when considering the streaming of scalable video to overlays consisting of terminals with different characteristics and physical limitations with respect to upload/download bandwidths (e.g. mobile devices).

The sixth challenge for P2P streaming systems is to become network aware, to take the underlying network's characteristics into consideration, while at the same

time maintaining a good performance. Construction of purely random overlays can cause problems to Internet Service Providers (ISP) or other managing entities of Autonomous Systems (AS) where the peers of the overlay reside, by incurring large costs to them due to the amount of traffic originating from or addressed to their networks. Network awareness demands the construction and maintenance of the overlay in a way that takes the AS boundaries into account and that minimizes the amount of costly inter-AS traffic. However, overlay construction based on proximity criteria imposes restrictions on the paths over which data can be diffused in the overlay, and may deteriorate the streaming performance compared to randomly formed overlays. It is therefore necessary to devise proximity-based overlay construction techniques that can maintain high streaming performance.

Chapter 3

Architectures

Any P2P streaming system can be decomposed into two sub-systems that work in parallel: overlay construction and maintenance, and chunk forwarding. The overlay construction and forwarding algorithms constitute the P2P protocol. Depending on the design choices with respect to the overlay construction and chunk forwarding, P2P systems with different characteristics can be built.

3.1 Tree-based streaming

The first overlay architectures proposed for P2P streaming were based on a single tree. In this case all peers are organized in a tree rooted at the server node, which is the source of the stream. Each node in the tree can have as many children as its capacity relative to the streaming rate allows. The tree construction and the parent-child relations can be determined by factors such as the end-to-end latency between peers [9], available bandwidth, or the underlying physical topology [10]. The video is segmented at the source into data units called packets or chunks. We will refer to them as packets, but in order to avoid confusion, we emphasize that, by that term, we refer to the application layer data unit. A packet is sent from the source to its children in the first level of the tree and they, in turn, push it to their children until all the peers in the tree receive the packet. Though it is a structure simple to construct, the single tree architecture has many drawbacks. First, few peers bear the load of forwarding, while most of the peers are leaf nodes and do not contribute at all to the overlay. Second, a peer that does not have upload capacity at least as high as the streaming rate cannot be used for forwarding, and has to become a leaf node. This leads to suboptimal utilization of the available resources. Third, the single tree structure is vulnerable to peer departures. When a peer in the higher levels of the tree departs, all of its descendants, that is the peers in the subtree rooted at the departed peer, will be cut off from the overlay and will not receive anything until they are reconnected to the tree. Fourth, the depth of the overlay can become large, which leads to peers in the last levels having a large

playback lag compared to the ones in the levels close to the root.

To increase the resilience of the overlay and to improve the capacity utilization, multiple-tree based solutions have been proposed. In a multiple-tree based system peers are organized in an overlay consisting of more than one trees. Each peer becomes member and receives data in all trees. The server is the root of all trees; a peer is assigned a different parent in each tree. A peer is an internal node and forwards data in some of the trees and is a leaf node and only receives data in the rest of the trees. The stream is divided into sub-streams and different parts of the stream are sent down different trees. In the following we present the general characteristics of a multiple-tree based overlay.

In multiple-tree overlays, the tree construction has to fulfil two requirements. First, it has to exploit path diversity in delivering data [11], that is, peers should receive data through different overlay paths in different trees, alleviating thus the effect of peer departures. Second, a tree construction algorithm has to create trees that are as shallow as possible. This requirement is significant since the number of levels of the tree is related to the playback delay at the peers. Peers that are situated far away from the root experience large delays compared to the ones that are high up in the tree. The fulfillment of these requirements is especially challenging under churn.

In the first multiple-tree construction algorithm proposed in [12], each peer becomes member of all trees, and receives and forwards data in each tree. Whenever a peer needs to be inserted in a tree, the overlay construction algorithm returns the peer with available capacity situated closest to the root. Available capacity refers to the ability of a peer to forward the sub-stream in that tree. Since by using this algorithm, peers have one child per tree, these trees are called minimum breadth trees. An example of such an architecture for a small overlay of 12 peers, organized in 3 trees and with the server having 3 children per tree, is shown in Fig. 3.1. The minimum breadth tree architecture offers the advantage of easy maintenance, even in periods of high churn. However, the depth of the formed trees grows linearly with N, the number of peers in the overlay. Due to the long paths from the source to all peers, a peer departure affects on average many peers, even if the time needed for tree reconstruction is very short. Furthermore, the playback delay for peers that are in the last levels of the overlay can be large, which is undesirable for a streaming architecture.

Lower delays and high diversity can be achieved when peers are internal nodes with many children in one tree and leaf nodes in the others. When peers have all their children in one tree, trees in the overlay have the minimum possible depth, therefore they are referred to as minimum depth trees. An example of a minimum depth tree overlay is shown in Fig. 3.2, where we show an overlay of 12 peers, organized in 3 trees and with the server having 3 children per tree. In minimum depth trees the depth of the trees is $O(logN)$, which yields shorter average playback delay and better scalability in the number of peers, compared to the minimum breadth trees. Peers forward the sub-stream that they receive in the tree where they are internal nodes. In the rest of the trees they only receive data but do not

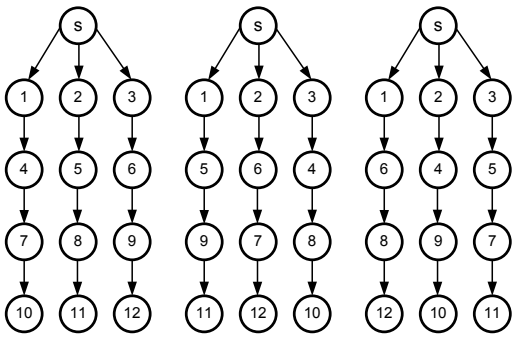


Figure 3.1: A minimum-breadth tree overlay of 12 peers organized in 3 trees and with the server having 3 children per tree.

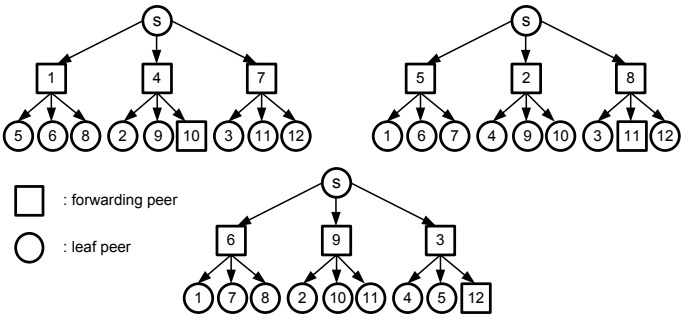


Figure 3.2: A minimum-depth tree overlay of 12 peers organized in 3 trees and with the server having 3 children per tree.

forward them. Such overlays were initially introduced in [12] and [13]. In the first work the construction of the overlay is performed by a centralized node, while in the latter it is coordinated by a distributed protocol.

The overlay construction and maintenance under churn for minimum depth overlays is more involved than the one of minimum breadth trees. Maintaining the overlay under churn introduces two challenges. The first challenge is to maintain a balanced capacity allocation over all trees. When a peer joins the overlay, it increases the capacity in the tree where it is forwarding and decreases the capacity in the trees where it is only receiving. Therefore it is not necessary that all trees have the same capacity at all times, even though trees have the same number of

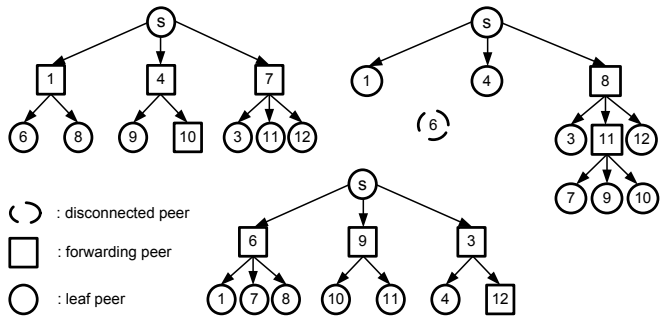


Figure 3.3: Disconnection of peers after the consecutive departures of internal nodes from the same tree of the overlay.

peers. Similarly, when an internal node departs from a tree, there might not be enough capacity to reconnect all its children in that tree, leading thus to some peers being disconnected. This phenomenon is depicted in Fig. 3.3, where after the departure of peers 2 and 5, which were forwarding in the second tree, there is not enough capacity for peer 6 to be connected to the tree. At the same time, there is an excess of capacity in the first and third trees that could be used to serve peer 6. Disconnected peers have to remain in this state until a new forwarding peer arrives in that tree or some leaf peer departs. Alternatively, the overlay has to be re-organized to balance the available capacity among the trees. The streaming quality is affected in both cases. Therefore, one of the targets of the construction algorithm is to keep the available capacity balanced over all trees.

The second challenge of the tree maintenance algorithm is to keep the trees at all times as shallow as possible. Whenever a peer needs to be connected, or a subtree reconnected, the tree construction algorithm has to locate the available position that is closest to the source of the stream. When trees are optimally maintained, the leaf nodes are in the same level at each tree in the overlay. Optimal tree maintenance requires reorganizing the overlay at each peer departure. After the departure of an internal peer, its children along with their descendants have to be reconnected to the tree. Optimal maintenance with respect to the depth of the tree suggests that each peer in the disconnected subtrees, rooted at the children of the departed peer, issues a separate reconnection request and is individually reconnected to the tree [14]. However, this incurs a large overhead, specially when the departed peer is high up the tree. Therefore a more practical method that sacrifices tree optimality for lower overhead reconnects only the children of the departed peer to the tree, while the whole subtree below them remains unchanged [15].

3.2 Mesh-based streaming

Considering churn as the main factor that impedes seamless streaming, mesh-based systems focus on creating an overlay that adapts fast to network changes. The rationale is that peers must not waste time in maintaining the overlay and peer departures should be dealt with promptly. So, the overlay construction in mesh systems is simple and fast. This property of the mesh-based systems, though, has an impact on the forwarding behavior of the peers. Lacking a rigid structure, sophisticated forwarding algorithms are needed at the peers.

This reveals the contrast between tree-based and mesh-based systems. In the former, the overlay construction is more sophisticated and costly in terms of both time and complexity, while data forwarding is simple. Whereas in the latter, building the overlay is simple, inexpensive and fast but forwarding is more complex. In this section we discuss the characteristics of the main components of a mesh-based system: overlay construction and data forwarding.

Overlay construction

In a mesh overlay, the goal of a peer is to maintain a large enough number of incoming connections, so that it is not affected by potential neighbor departures. The overlay is usually built in a distributed way and each peer is only aware of a small subset of participants, which constitutes its neighbors. The neighbor selection policy differs from system to system but the intent is to keep the neighbor selection process simple and fast. The simplest policy to select neighbors is to do it at random. A peer receives a randomly generated list of peers already connected to the overlay and tries to establish connections to them. This approach, though, does not take into account any parameters of the underlying network topology that could optimize the data exchange. More complex neighbor selection policies base the establishment of neighbor relationships on criteria such as available bandwidth and latency [16].

Data distribution

In contrast to tree architectures, forwarding in mesh overlays is not straightforward. The lack of a concrete structure that defines the flow of data from a peer to its neighbors makes it impossible to determine in advance the path through which data is diffused. As also done in tree-based systems, in mesh-based systems packets are sent from the server and then relayed by the peers. Packet forwarding is based on decisions taken locally at the peers. These decisions are based on information about the availability of packets in neighboring peers.

In mesh overlays, data packets can be exchanged following the push or the pull approach. In either case a peer has to make a decision of which packet to send to which neighbor (push scheme), or which packet to ask from which neighbor (pull scheme). Both approaches have their pros and cons. A push approach is expected

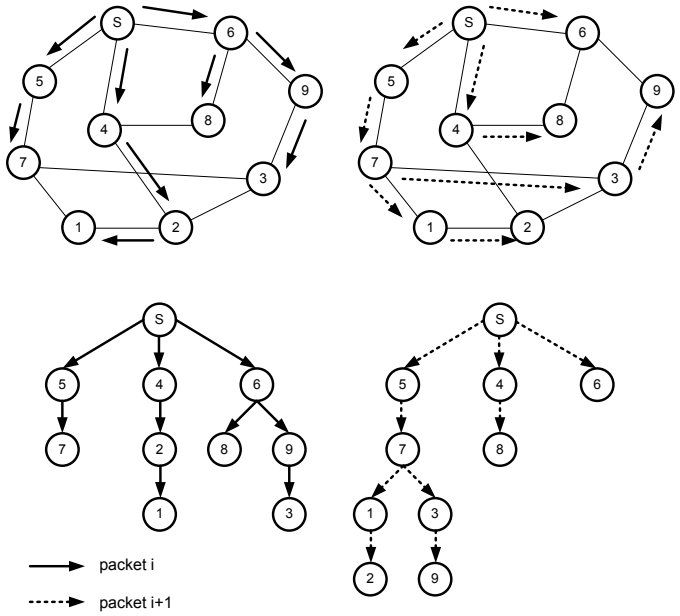


Figure 3.4: Spanning trees that are formed for the packet delivery of two packets in a mesh overlay consisting of 9 peers

to work better for uplink constrained peers, since it avoids multiple requests for packets at a peer. On the other hand, a pull approach is a good choice for downlink constrained peers since a peer can control the incoming rate of packets from its neighbors. For both schemes however, the lack of coordination among peers that characterizes such a distributed environment, creates inefficiencies in the data propagation. In a push based system, this is translated to multiple copies of packets sent to the same peer by its neighbors. In pull based systems, a peer can be flooded with a number of packet requests from its neighbors exceeding the maximum number of requests that it can serve. Moreover, pull based systems have higher overhead since data packets have to be explicitly requested through control packets. These phenomena lead to suboptimal performance of both forwarding schemes.

Regardless of the forwarding logic employed, each packet is sent out from the server and it reaches all peers by forming a spanning tree of the overlay graph. Since the forwarding decisions are taken locally at each peer, the peer degree and depth of the tree for each packet cannot be known a priori. Moreover, the trees that are formed for each packet can be very different, as depicted in Fig. 3.4, where we plot the paths for two packets in a stream over an overlay of 9 peers.

3.3 Hybrid systems

Hybrid systems [17, 18] were introduced in an effort to combine the advantages of tree-based and mesh-based P2P streaming systems. These systems try to combine the robustness of the overlay offered by the mesh-based systems with the simple and efficient forwarding over a fixed structure that tree-based systems provide. In hybrid push-pull systems, the stream is split into a number of sub-streams like in the tree-based systems. For a peer to receive at full streaming rate, it has to find parents that can feed it with all the sub-streams. Upon joining the overlay, each peer receives a list of already joined peers that can act as its suppliers. It then tries to find among these peers a supplier peer for each of the sub-streams. After a peer locates a parent that can provide it with a sub-stream, it requests (pulls) from that parent a packet belonging to the substream. After the initial pull request, the parent keeps feeding the peer with subsequent packets of the sub-stream, according to the push paradigm employed in the tree-based systems.

Chapter 4

Performance evaluation

4.1 Data forwarding in P2P streaming systems

Forward error correction in multiple-tree-based systems

Packets disseminated over an overlay path can be lost due to congestion in the underlying physical network, or due to the departure or failure of intermediate peers. The loss of a packet on an overlay link does not only affect the recipient peer at the end of that link but also all of its descendants. As the loss propagates to the levels below the one where it occurred, it reduces the stream quality for a potentially large fraction of peers.

One way to deal with loss propagation in an overlay is through the use of retransmissions [19]. Whenever a peer perceives loss of packets in its incoming link it requests retransmission of the missing packets by its parents. This solution, though, can make it difficult to satisfy the stringent delay constraints that streaming applications impose and therefore has not received much attention. The other way of protecting the overlay against loss propagation is through the use of channel coding, or joint source-channel coding schemes. Implementation of such coding schemes seems to fit naturally to multiple-tree based overlays, due to the path diversity that they offer. These schemes include layered coding, multiple description coding (MDC) and forward error correcting codes (FEC). The use of such schemes in multiple-tree based overlays is based on the expectation that a peer will be able to partially or completely recover data that was lost on the path from the source to itself, as long as the peer is not directly connected to the congested physical link.

In the case of layered coding (for an overview of the scalable extension of H.264/AVC see [20]) the video is segmented and encoded into a base and a number of enhancement layers. The base layer provides a basic level of quality and each enhancement layer - successfully received and decoded - enhances the quality of the stream. Without the reception of the base layer enhancement layers are useless. When the video is segmented into layers then each layer can be error protected using redundant information proportional to the significance of the layer [21]. Using

MDC (for an overview see [22]), the video is coded into descriptions, where each description can be independently decoded and has more or less the same contribution to the decrease in the video distortion. The more descriptions a peer is able to receive, the higher its video quality is going to be. MDC has been combined with multiple-tree architectures in order to overcome errors in the overlay, provide resilience to churn and to account for peers that do not possess the adequate bandwidth to receive at the full streaming rate [13, 23].

When FEC is employed, e.g. Reed-Solomon codes [24], data is segmented into packets and for each k packets another c redundant ones are added in order to form a block of $n = k + c$ packets. When used with multiple-tree based streaming, packets of a block are sent down the trees in a round-robin way. If a peer receives any k out of the n packets in a block, it can reconstruct the rest. It has been shown that for any loss probability over an overlay link there is a level of redundancy that can ensure delivery of data to tree levels arbitrarily far away from the server with a probability greater than zero [25, 26].

In this thesis we extend [25, 26] to study the data distribution in a generalized multiple-tree overlay using forward error correction and assuming homogeneous and independent link losses. In paper A, we present a model that derives the packet possession probability of an arbitrary peer as a function of the level where it is situated in its forwarding trees. We identify the existence of a stability region with respect to the link loss probability. Within this region, the packet possession probability remains high and is not affected by the number of forwarding trees. Outside the stable region though, the performance degradation is severe and the performance is negatively affected by the number of forwarding trees of a peer. Moreover, we derive asymptotic bounds on the packet possession probability when the overlay consists of a very large number of peers. We also conclude that the standard deviation of the received packets in a block is an adequate measure to control the adaptation of the FEC to the overlay conditions.

Data distribution in mesh-based systems

The non-deterministic tree structure that is formed for the delivery of each data packet to the peers creates two issues worth exploring. First, to determine whether a packet can be delivered to all peers using a particular forwarding algorithm. Second, to find an upper bound on the time it takes from the moment the packet is sent out from the source until it reaches all peers. A perfect forwarding algorithm guarantees the reception of a packet by all peers and, for a given overlay, gives a deterministic bound on the time to deliver a packet to all peers [27]. However, such an algorithm requires coordination among all peers and is thus not feasible due to the distributed nature of forwarding in mesh-based systems. Instead, the efficiency of a realistic forwarding algorithm in delivering the stream on time is characterized by the trade-off between playback delay and playout continuity.

Numerous works address the issue of data propagation in mesh-based overlays by proposing and evaluating different scheduling algorithms. In [28] it is shown that

optimal streaming is possible in fully connected meshes and the authors propose scheduling algorithms for the cases when the bottleneck is at the uplink or at the downlink of the peers. In [29], the authors derive the playback delay–continuity curves for various push forwarding algorithms assuming a fully connected mesh and perfect coordination among the peers. The optimality of a class of deadline-based algorithms is theoretically proven in [30], where the authors also investigate through simulations the impact of the neighborhood size of peers on the streaming performance. In [31, 32, 33], the authors suggest a combination of push and pull approaches to coordinate the packet exchange and to avoid reception of duplicate data packets or a flood of request packets at the peers.

For the sake of simplicity and tractability, analytical approaches to data distribution in mesh-based systems assume content encoded at a constant bit-rate and evaluate the efficiency of data distribution via the playout miss ratio. It is, thus, implicitly assumed that the loss of any given packet has the same impact on the video quality at the peers. The modern encoding standards though, produce variable bit-rate videos comprising of different frame types, whose contribution to the video quality in terms of PSNR is different. Oblivious packetization of the video leads to worse performance compared to the case when each packet contains video frames corresponding to the same group of pictures [34]. Forwarding schedulers that take into account the video characteristics are called video aware schedulers. Using video aware schedulers, the live streaming system performance can be increased by prioritizing packets containing video frames whose playout miss would have a severe impact on the PSNR of the played out video [35, 36].

Besides the analytic and simulative approaches, an insight on the performance of live P2P streaming systems is also obtained by carrying out measurements on existing systems. In the recent years, there have been numerous popular P2P streaming applications that were able to attract millions of viewers. Most of these applications are proprietary and hence, measurement studies are the only way to evaluate their performance, including the effect of parameters that are not easy to evaluate analytically. Measurement studies help us to obtain an understanding of the peer membership and forwarding logic of these applications and one can see different approaches as far as forwarding is concerned. In [6], the authors conclude that there are applications, such as *PPLive*, where peers receive the video mostly from a small set of peers, while in other applications (e.g. *SopCast*), peers receive data from many other peers at the same time. Furthermore, measurement studies enable us to evaluate the performance of streaming systems with respect to metrics that cannot easily be captured via analytical approaches, such as startup delays. In [7], the authors find that startup delays can vary from some seconds to a couple of minutes depending on the stream popularity, which reveals a major problem of P2P streaming in general.

In this thesis we investigate the trade-off of playback delay versus playback continuity by building a general framework for evaluating different forwarding schemes. In paper B, we propose a model to describe the data propagation in an overlay consisting of peers with an upload capacity equal to the stream rate. We assume that

Table 4.1: Indicative upload bandwidth distribution for a streaming event [14].

Type	**Degree**	**Number of hosts**
Free-riders	0	58646 (49.3%)
Contributors	1	22264 (18.7%)
Contributors	2	10033 (8.4%)
Contributors	3-19	6128 (5.2 %)
Contributors	20	8115 (6.8 %)
Unknown	-	13735 (11.6 %)
Total	-	118921 (100%)

the overlay is a regular graph and there is no coordination among peers. Based on the model and simulations we derive the playback delay–continuity trade-off curve for various forwarding algorithms and show that random packet forwarding exhibits good delay characteristics and scalability. Moreover, we investigate the impact of the number of neighbors of a peer and conclude that it is positively correlated to the playout probability for small values but after it reaches a threshold value, it no longer affects the playout probability. We also deal with the issue of the imperfect information at the peers and show that even small delays in the control information exchange lead to fast increase of the playout missing ratio.

4.2 Streaming in heterogeneous P2P networks

For the sake of simplicity, in the analysis of data distribution in P2P streaming systems, it is often assumed that peers are homogeneous in terms of upload bandwidth. This is however far from the truth in the current Internet. There exist peers whose upload capacity can be many times higher than the streaming rate, others with upload capacities a couple of times the streaming rate and a considerable fraction of peers with upload capacities lower than the streaming rate or no upload capacity at all. As an indication of the bandwidth heterogeneity, in Table 4.1, we show results obtained by a measurement study corresponding to a large streaming event consisting of more than 100000 peers [14]. The degree of each peer category is defined as the ratio of upload bandwidth over the stream encoding rate. The very large number of low and non-contributing peers can be partly explained by the asymmetric nature of access technologies for many residential users. Users connected through ADSL lines have fast downlink, enabling them to watch a video at a high encoding rate, but not allowing them to forward it to other peers at this rate. Furthermore, peers behind firewalls and NAT-enabled routers constitute another factor that could explain these figures. Although, there have been many firewall and NAT traversal schemes proposed ([37, 38] and references therein), they have not been widely implemented.

As already stated in Section 2.2, the bandwidth heterogeneity raises two chal-

lenges for P2P streaming systems: maintain streaming feasibility and achieve low delay streaming. Feasibility depends on the sum of available upload capacities in the overlay with respect to the streaming rate. The delay depends on the depth of the trees in the overlay. The shallower these trees are, the lower is the average playback delay.

Streaming feasibility through overlay construction

For a given distribution of peer upload bandwidths, minimum delay streaming occurs when data is diffused to the peers in descending order with respect to upload bandwidth [39]. That is, peers with higher upload rates receive packets earlier than peers with lower ones. Therefore, to minimize the delay for an individual tree would require a fast peer being able to replace a slower one at its position in the tree structure [40, 41]. The slower peer is then reconnected as a child of the faster one or as a child of another peer with available capacity. We refer to the replacement of a peer in a tree structure by a faster peer as preemption. When considering a multiple-tree overlay as a whole though, preemption in a tree must not be based on the contribution in one specific tree only. In [40], the authors show, using simulations based on real and synthetic traces, that preemption should depend on the contribution of a peer in all trees and not on its contribution in an individual tree. By pushing the low contributing peers towards the last levels of the distribution trees, a P2P system can achieve streaming feasibility, at least for the high contributing peers, as well as low playback delays.

A similar approach can be followed by mesh-based systems as well. Maintaining the spanning trees as shallow as possible, by hierarchically placing the peers in the trees based on their upload bandwidth, is not a trivial task, since the shape of the trees over which packets are forwarded depend on local decisions taken at the peers and is not known a priori. However, several works have dealt with the issue of low delay streaming in mesh overlays. Lobb et al. [42] propose a low delay P2P-TV system where peers with high contribution are pushed closer to the server through the increase of their neighborhood size, improving thus the delay performance of the overlay. A similar approach is presented in [43], where peers select their parents based on the measure of *power*, which takes into account both the bandwidth of the peer as well as the latency between the two peers.

At the same time the video quality at the peers can be increased by combining the above preemption/prioritization schemes with transmission of scalable video [20]. The video is encoded in layers so that peers that cannot receive/forward at the full streaming rate and are pushed further away from the source, are still able to receive the stream at a certain quality [44, 45].

In the case of non-scalable video, when the upload capacity is not enough to stream to all peers, there can be two options for providing seamless streaming to all peers at a given rate. The first one is to lower the streaming rate to a sustainable one given the available upload rate in the overlay. The second option is to implement an admission control policy that does not allow the demand to exceed the available

supply of resources in the overlay for a given streaming rate. Both schemes require fast and efficient adaptation to the dynamic network conditions.

In this thesis, we address the issue of providing seamless streaming in dynamic P2P overlays through admission control. In paper C, we consider the problem of providing seamless streaming at a given rate to all peers in an overlay consisting of contributing and non-contributing peers. We develop a model that describes the evolution of the overlay capacity over time. We propose two admission control schemes that block or drop non-contributing peers when the overlay resources decrease to a point where streaming to all peers becomes infeasible. We implement our scheme in a multiple-tree overlay and show that it leads to high resource utilization, while at the same time it ensures that, once admitted, peers experience high streaming quality.

Streaming feasibility through incentives

According to [39], faster peers have to receive packets earlier than slower ones in order to achieve minimum delay streaming. Several incentive schemes have thus been proposed that reward high contributing peers by placing them close to the streaming server. For tree-based systems this is a rather easy task, especially if the overlay construction is taken care of centrally. In [23], the authors consider a multiple-tree-based overlay consisting of peers with various capacities ranging from zero contribution (free-riding) up to many times the streaming rate. A taxation scheme is proposed so that peers with high capacities, contribute more than they receive in order to make streaming feasible to all peers. An incentive scheme that relates a peer's contribution to its received stream quality is presented in [46], where peers maintain a number of backup parents that is proportional to their contribution. A peer can switch parents if receiving the stream from a backup parent incurs a lower playback delay than the one it currently perceives. Thus, under this scheme peers that contribute more, receive the stream at a low delay with high probability.

Maintaining a contribution level adequate to sustain streaming to all peers is not an easy task in a mesh-based system. The absence of a centralized monitoring scheme can neither guarantee nor enforce peer cooperation through the implementation of an admission control scheme, in contrast to a tree based solution. Therefore, most of the proposed solutions to enforce cooperation are implemented in a distributed way by the peers. The majority of these schemes is inspired by the incentive mechanisms for P2P file distribution and relate the streaming quality of a peer to the peer's contribution. In [47], a rank-based tournament is proposed, where peers are ranked according to their total upload contribution and each peer can choose as neighbor any peer that is below itself in the ranked list. Thus, peers that have high contribution have also higher flexibility in selecting their neighbors. In [48], the authors propose a payment-based incentive mechanism, where peers earn points by uploading to other peers. The supplier peer selection is performed through first price auctions, that is, the supplier chooses to serve the peer that of-

fers her the most points. This means that peers that contribute more, accumulate more points and can make high bids, thus increasing their probability of winning an auction for a parent. Differentiated streaming quality can be obtained by using joint source-channel coding techniques and a distributed incentive mechanism on the peers, as in [49] with layered video coding and in [50] using multiple description coding.

The incentive mechanisms proposed in the literature start, to a large extent, from the assumption that peers have the ability to contribute but refrain from doing so. However, peers might be unable to have a substantial contribution because of their last-mile connection technology instead of their unwillingness to contribute. Most DSL and cable Internet connections are asymmetric, hence peers may have sufficient capacity to, e.g., download high definition video but insufficient capacity to forward it. Similarly, in high-speed mobile technologies, such as High Speed Downlink Packet Access (HSDPA), the download rates are an order of magnitude higher than the upload rates [51]. Incentive schemes that relate a peer's contribution to its streaming quality, fail to provide adequate quality for peers that have limited contribution. It is reasonable, though, for one to argue that these peers should not be denied service by the overlay as long as there is available capacity to serve them as well.

In this thesis we study the maximization of the streaming performance in heterogeneous P2P mesh-based overlays through the use of incentives. In paper D, we consider overlays consisting of contributing and non-contributing peers. We model the behavior of the contributing peers with respect to their generosity against their non-contributing neighbors and study the social welfare of the system. We show that when contributors are selfish the social welfare is sub-optimal. Therefore, we propose an incentive mechanism to maximize the social welfare, which uses the server's capacity as an incentive for contributors to upload to non-contributing peers as well. We prove that our mechanism is both individually rational and incentive compatible.

4.3 Playout adaptation in P2P systems

As the overlay increases in size, the playback delay of the peers has to be increased as well in order to maintain the same playout miss ratio [52]. At the same time, several measurement studies on P2P streaming systems reveal abrupt transitions from small overlays to large ones at the beginning of live events [53, 54]. These transitions, called *flash crowds* [55, 56], can lead to the expansion of the overlay size in terms of orders of magnitude. It is therefore paramount that the P2P system has a playout adaptation algorithm that can progressively adapt the playback delay of the peers to the new overlay size.

Playout adaptation is the act of changing the playback delay of the peers. It can be achieved in two ways. First, by pausing the playout of frames in case of underflow, allowing thus the buffer level to build up again, or by dropping frames in

the case of overflow [57, 58, 59]. Second, by altering the frame duration with respect to the original encoded video sequence [60, 61]. When an underflow is about to occur, frames are played out at a decreased rate compared to the nominal video rate. On the contrary, when an overflow is about to occur, frames are played out at a rate higher than the nominal video rate. Playout adaptation via changing the playout rate at the end hosts is enabled by several models showing that slight changes in the video rate have only a minor impact on the perceptual video quality [62, 61].

In this thesis we use playout adaptation in P2P live video streaming systems in the form of altering the frame duration at the peers in order to improve the performance of video delivery in overlays under churn. In Paper E, we propose two playout adaptation algorithms: one coordinated and one distributed. The algorithms tune the playback delay of the peers in the overlay so that the playout miss ratio is maintained within a predefined interval. We validate the performance of the algorithms under different churn models and show the benefits from their use compared to overlays where no playout adaptation is used in terms of distortion of the received video.

4.4 Network awareness

Autonomous Systems (ASs) in the Internet are managed by different entities (Internet Service or Content Providers) and there is an associated cost with traffic exchange between two ASs. The cost for traffic exchange between two ASs depends on their interconnection, which can be of two types: peering and transit provider-customer. When two ASs have a peering agreement, then the traffic exchanged between them is for free, as long as it is approximately balanced. When an AS provides transit to another AS, the customer AS has to pay for the traffic routed through the provider's AS, usually according to the 95-percentile rule, that is the client AS is billed for the 95% of the maximum traffic rate observed over the charging period.

The cost of the traffic generated by a P2P system depends on the distribution of the overlay traffic to the different types of physical links in the underlying network. From the point of view of an ISP, traffic routed through a transit link is more expensive than traffic routed through a peering link, which in turn is more expensive than traffic that stays within the ISP boundaries. Peers in a P2P overlay can span a wide geographical area and be distributed over a large number of ASs. A randomly created P2P overlay can generate a large amount of traffic routed through transit links, incurring large costs to the ISPs. Measurements of offline P2P file sharing showed that in random overlays peers download blocks of data through expensive links even though the data was available in peers residing in the same ISP at no cost [63]. The need for network awareness in overlay construction became even more apparent when ISPs started throttling the traffic generated by P2P networks [64].

The construction of a network-aware overlay is based on information that the P2P application receives about the network over which it is deployed. Information

can be in the form of ranking of candidate neighbors with respect to a variety of locality criteria: residency in the same AS, AS-hop distance in the BGP path and network latency [65]. Network information can also be in the form of cost associated with the selection of certain paths in the underlying network, according to criteria chosen by the ISP [66]. Such information about the network can be obtained by a dedicated infrastructure run by a third party or the ISP itself [65, 67, 68]. There is an ongoing effort within the IETF ALTO working group for the standardization of a protocol that enables the exchange of information about the network between the ISPs and the P2P application [69].

The investigation of network awareness-performance trade-off for offline content distribution shows that locality aware overlay construction can lead to peer clustering, reducing thus the data distribution performance of the P2P system [70, 65, 68, 71]. Peer clustering has a negative impact on the performance of P2P streaming systems as well [72, 73]. Therefore, the level of network awareness must be carefully selected so that there is a balance between the associated traffic cost and the data distribution performance. It is usually necessary to maintain a number of expensive connections in the overlay in order to maintain high streaming performance. Thus, network aware neighbor selection in P2P streaming systems consists in a weighted selection of peers from disjoint sets of potential neighbors, ordered according to locality (cost) or capacity criteria [72, 73, 74, 36].

Most of the works on network aware overlay construction distinguish the overlay connections in intra-AS and inter-AS only, neglecting the fact that there are two substantially different types of inter-AS connections with respect to traffic cost. In this thesis we address the problem of constructing network aware overlays by making a clear distinction between overlay connections between peers residing in ASs that have a peering agreement and overlay connections where P2P traffic has to cross a transit link. Furthermore, we take into account recent measurement studies that reveal a transformation of the Internet towards a flatter structure due to the increasing number of peering agreements between ISPs [75, 76], opening up thus new possibilities for network-aware streaming. In paper F, we use as network awareness metric the amount of P2P traffic routed through *transit inter-AS* links and devise overlay construction policies that maximize the streaming performance in the overlay for a given network-awareness level.

Chapter 5

Summary of original work

Paper A: Streaming Performance in Multiple-tree-based Overlays

György Dán, Viktória Fodor and Ilias Chatzidrossos. In Proceedings of IFIP Networking, Atlanta, Georgia, May 2007.

Summary: In this work we evaluate the data transmission performance of the generalized multiple-tree-based overlay architecture for peer-to-peer live streaming that employs multipath transmission and forward error correction. We give mathematical models to describe the error recovery in the presence of packet losses. We evaluate the data distribution performance of the overlay, its asymptotic behavior, the stability regions for the data transmission, and analyze the system behavior around the stability threshold. We argue that the composed measure of the mean and the variance of the packet possession probability can support adaptive forward error correction.

My contribution to this work was the construction of a simulator and the simulation based performance evaluation of the data transmission using FEC under a multiple-tree based architecture.

Paper B: Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates

Ilias Chatzidrossos, György Dán and Viktória Fodor. In P2P Networking and Applications, vol.3, num.1, March 2010.

Summary: In mesh-based peer-to-peer streaming systems data is distributed among the peers according to local scheduling decisions. The local decisions affect how packets get distributed in the mesh, the probability of duplicates and consequently, the probability of timely data delivery. In this paper we propose an analytic framework that allows the evaluation of scheduling algorithms. We consider four solutions in which scheduling is performed at the forwarding peer, based

on the knowledge of the playout buffer content at the neighbors. We evaluate the effectiveness of the solutions in terms of the probability that a peer can play out a packet versus the playback delay, the sensitivity of the solutions to the accuracy of the knowledge of the neighbors' playout buffer contents, and the scalability of the solutions with respect to the size of the overlay. We also show how the model can be used to evaluate the effects of node arrivals and departures on the overlay's performance.

My contribution to this work was the development of the analytical model for the data propagation in the mesh-based overlay as well as the performance evaluation through modelling and simulations. The paper was written in collaboration with the two co-authors.

Paper C: On the effect of free-riders in P2P streaming systems

Ilias Chatzidrossos and Viktória Fodor. In Proceedings of the 4^{th} Telecommunication Networking Workshop on QoS in Multiservice IP Networks, pages 8-13, Venice, Italy, February 2008.

Summary: Peer-to-peer applications exploit the users willingness to contribute with their upload transmission capacity, achieving this way a scalable system where the available transmission capacity increases with the number of participating users. Since not all the users can offer upload capacity with high bitrate and reliability, it is of interest to see how these non-contributing users can be supported by a peer-to-peer application. In this paper we investigate how free-riders, that is, non-contributing users can be served in a peer-to-peer streaming system. We consider different policies of dealing with free-riders and discuss how performance parameters such as blocking and disconnection of free-riders are influenced by these policies, the overlay structure and system parameters as overlay size and source upload capacity. First we present a model that describes the performance of optimized overlays under the different policies, then we evaluate via simulations how more rigid overlay structures such as multiple-trees behave under the same conditions. The results show that while the multiple-tree structure may affect the performance free-riders receive, the utilization of the transmission resources is still comparable to that of an optimized overlay.

My contribution to this work was the modelling of the available capacity in the overlay as well as the performance evaluation of the proposed schemes through the use of analytical modelling and simulations. The main part of the paper was written by the second author.

Paper D: Server Guaranteed Cap: An incentive mechanism for maximizing streaming quality in heterogeneous overlays

Ilias Chatzidrossos, György Dán and Viktória Fodor. In Proc. of the 9^{th} IFIP Networking, Chennai, India, May 2010.

Summary: We address the problem of maximizing the social welfare in a peer-to-peer streaming overlay given a fixed amount of server upload capacity. We show that peers' selfish behavior leads to an equilibrium that is suboptimal in terms of social welfare, because selfish peers are interested in forming clusters and exchanging data among themselves. In order to increase the social welfare we propose tunable scheduling algorithms and a novel incentive mechanism, Server Guaranteed Cap (SGC), that uses the server capacity as an incentive for high contributing peers to upload to low contributing ones. We prove that SGC is individually rational and incentive compatible. We also show that under very general conditions, there exists exactly one server capacity allocation that maximizes the social welfare under SGC, hence simple gradient based method can be used to find the optimal allocation.

My contribution to this work was part of the system definition, the development of the model for the data propagation in the heterogeneous overlay and the performance evaluation of the proposed system through simulations. The paper was written in collaboration with the two co-authors.

Paper E: Playout adaptation for P2P streaming systems under churn

Ilias Chatzidrossos and György Dán. Submitted to Packet Video Workshop (PV) 2012.

Summary: We address the problem of playout adaptation in peer-to-peer streaming systems. We propose two algorithms for playout adaptation: one coordinated and one distributed. The algorithms dynamically adapt the playback delay of the peers so that the playout miss ratio is maintained within a predefined interval. We validate the algorithms and evaluate their performance through simulations under various churn models.pay We show that playout adaptation is essential in peer-to-peer systems when the system size changes. At the same time, our results show that distributed adaptation performs well only if the peers in the overlay have similar playback delays. Thus, some form of coordination among the peers is necessary for distributed playout adaptation in peer-to-peer streaming systems.

My contribution to this work was part of the problem formulation, the development of the simulation environment based on a publicly available simulator and the performance evaluation of the adaptation algorithms through simulations. The paper was written in collaboration with the second author.

Paper F: Small-world Streaming: Network-aware Streaming Overlay Construction Policies for a Flat Internet

Ilias Chatzidrossos, György Dán and Arnaud Legout. Submitted to International Conference on Distributed Computing Systems (ICDCS) 2012.

Summary: Recent measurements indicate that the peering agreements between Autonomous Systems (AS) are flattening the AS level topology of the Internet. The transition to a more flat AS topology opens up for new possibilities for proximity-aware peer-to-peer overlay construction. In this paper we consider the problem of the construction of overlays for live peer-to-peer streaming that leverage peering connections to the maximum extent possible, and investigate how a limited number of overlay connections over transit links should be chosen such as to maximize the streaming performance. We define a set of transit overlay link establishment policies that leverage topological characteristics of the AS graph. We evaluate their performance over regular AS topologies using extensive simulations, and show that the performance difference between the policies can be up to an order of magnitude. Thus, it is possible to maximize the system performance by leveraging the characteristics of the AS graph. Based on our results we also argue that the average loss probability is not an adequate measure of the performance of proximity-aware overlays. We confirm our findings via simulations over a graph of the peering AS topology of over 600 ASs obtained from a large measurement data set.

My contribution to this work was part of the problem formulation, the development of the simulation environment based on a publicly available simulator and the performance evaluation of the proposed policies through simulations. The paper was written in collaboration with the second author.

Publications not included in this thesis

- Gy. Dán, I. Chatzidrossos, V. Fodor and G. Karlsson,"On the Performance of Error-resilient End-point-based Multicast Streaming", in Proc. of 14th International Workshop on Quality of Service (IWQoS), pp. 160-168, June 2006
- Gy. Dán, V. Fodor and I. Chatzidrossos, "On the performance of Multiple-tree-based Peer-to-peer Live Streaming", in IEEE Infocom, Anchorage, Alaska, May 2007
- V. Fodor and I. Chatzidrossos, "Playback delay in mesh-based peer-to-peer systems with random packet forwarding", in Proc. of International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST), pp. 550-555, September 2008
- V. Fodor and I. Chatzidrossos, "Playback delay in mesh-based Peer-to-Peer systems with random packet forwarding and transmission capacity limitation", in International Journal of Internet Protocol Technology, Vol. 3 , no. 4, pp. 257-265, December 2008
- Gy. Dán, N. Carlsson and I. Chatzidrossos, "Efficient and Highly Available Peer Discovery: A Case for Independent Trackers and Gossiping", in Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P), August 2011

Chapter 6

Conclusion and Future Work

This thesis presents an evaluation of the performance of P2P live streaming systems and proposes solutions to improve the system performance. In the following, we summarize the main contributions of this thesis and point out the most significant conclusions that we derived.

We evaluated the efficiency of data distribution in the two main P2P architectures, presented in Chapter 3. For multiple-tree-based systems, we studied the efficiency of data distribution in conjunction with the use of FEC. We showed the existence of a stability region and determined it as a function of the link loss probability between the peers, the number of levels of the trees and the redundancy of the channel coding. Within the stability region of the system, data can be distributed to all the peers while outside this region the performance degrades sharply. We also identified the variance of the packet possession probability as a good candidate for driving a control algorithm that adapts the redundancy of the channel coding in order to keep the system in the stable region. For mesh-based systems, we developed a general framework to evaluate the efficiency of various forwarding schemes. We compared four push-based forwarding algorithms and showed that while fresh data first algorithms achieve faster dissemination for small playback delays, random forwarding has better asymptotic properties as the playback delay increases. We also evaluated the sensitivity of forwarding algorithms to the level of control overhead and found that random forwarding can perform well even at decreased control overhead, in contrast to the fresh data first algorithms.

We addressed the challenge of maintaining streaming feasibility under churn and in the presence of non-contributing peers through the use of admission control. We proposed two admission control policies to limit the number of non-contributing peers in the overlay so as to maintain streaming feasibility. For idealized overlays we showed analytically that it is possible to serve non-contributing peers with small disruptions, while ensuring that high contributing peers receive at full streaming rate. We also showed through simulations that the upload bandwidth utilization under these policies remains high even when used in multiple-tree-based systems.

We addressed the challenge of efficient resource utilization in heterogeneous push-based streaming systems through the use of incentives. We identified selfish peer behavior as a factor that can lead to suboptimal streaming performance and proposed an incentive mechanism to enforce peer cooperation. We proved that the proposed mechanism is individually rational and incentive compatible and showed its effectiveness through simulations. Guaranteeing maximum streaming quality for the high contributing peers, the incentive mechanism combined with layered video coding allows the distribution of video at a reasonable quality to physically constrained, low contributing peers.

We addressed the challenge of maintaining good streaming performance under churn through the use of playout adaptation. We proposed two adaptation algorithms and showed the benefits obtained from using playout adaptation compared to overlays where no adaptation is used. Our results, together with the measurement studies showing the evolution of P2P streaming overlays during live events, indicate the importance of playout adaptation in modern P2P streaming systems. At the same time, we also identified a potential problem when performing playout adaptation in a distributed way and argued that distributed adaptation should be combined with a mechanism that maintains a minimum playback lag among the peers.

Finally, we investigated the potential of a novel approach to the construction of network aware overlays. We made the distinction between peering and transit *inter-AS* links and considered recent measurement studies that reveal a transformation of the AS-level topology of the Internet to a flatter structure, where peering agreements between ASs are becoming more and more common. Based on these findings, we devised overlay construction policies that maximize the streaming performance while making use of only a small number of expensive transit links. Our simulation results, on regular as well as on measured topologies, indicate that it is possible to construct better than random overlays based on topological information. The proposed overlay construction policies could therefore be used in conjunction with services that provide network information to P2P systems and lead to network aware P2P overlays that offer good streaming performance.

Future work

After several years of research and hands-on experience from a large number of deployed P2P systems, it is safe to say that P2P streaming is now well past its infancy. The fundamental properties of P2P live streaming systems have been analyzed and well understood by the community. As far as system architecture is concerned, mesh-based systems seem to have prevailed due to their robustness to churn and their performance is generally evaluated based on the playback delay-loss probability curve, used like the rate-distortion curve in the case of video coding. However, there are still open issues with respect to the optimal deployment of such systems over the Internet, some of which we addressed in the context of this thesis.

One of the most important open issues is the system scalability, that is how

a system can grow in size while maintaining seamless streaming for the already joined peers. Our coordinated algorithm for playout adaptation is a first step towards this direction. In coordinated adaptation peers are playing out the same chunk at any given time instant. When considering distributed adaptation though, the buffers of the peers may drift apart from each other, leading to a decreased efficiency in data exchange. In Paper E, we showed such a scenario for a distributed adaptation algorithm, when the overlay is shrinking in size after a live event. It is therefore necessary to provide a mechanism that maintains a high level of buffer overlap among the peers. Furthermore, in such systems it is worth investigating the impact of the peer startup process, that is, the selection of the initial chunk interest window of the peers, on the system performance. The choice of initial window implicitly defines the playback delay of the peers. Considering the trade-off between playback delay and playout miss ratio, it is also worth investigating the impact of strategic peer behavior on the system performance as peers might try to increase their playback delay in an uncoordinated way so that they achieve lower playout miss ratio.

Another implementation issue still not fully explored is the construction of network aware overlays. In spite of the significant work that has been conducted on that field during the last few years, there are still unanswered questions. First, it is not clear what is the optimal mixture of criteria for the peer ranking during the construction of the overlay, that is, the weight of locality and capacity aware neighbor selections. Especially for the case of locality criteria, one can look at different granularities of locality (AS-level, WAN-level, LAN-level). Besides the issue of neighbor selection criteria, it is also still not clear which entity is going to provide the information about the network to the P2P application. The current IETF ALTO draft states that this can be either a third-party or the ISP itself if the service is provided on a per-ISP basis. In the first case, ISPs may be reluctant to reveal information about their network infrastructure or to lose their sovereignity in terms of strategic decision making, making this solution problematic. In the latter case, one has to account for the strategic behavior of ISPs. For instance, a transit ISP has no incentive in enforcing locality awareness since this would reduce its revenue, given the fact that charges are according to the traffic volume. Consequently, the deployment of network information provision services calls for a game theoretic analysis of the behavior of the key entities involved, that is ISPs, content providers and P2P system operators.

Finally, given the research results thus far, P2P streaming does not appear as a solution that can replace traditional TV, or even IP-multicast based IPTV systems. Instead, it offers a perfect way of transmitting ad-hoc live events to potentially large audiences, or scheduled events with a high global interest such as conferences or workshops without any infrastructure investment. Nevertheless, it is probably not suited for stand-alone commercial use, as service guarantees cannot be given by the provider. In spite of that, P2P streaming has great potential as an auxiliary component of a hybrid multicast-broadcast system. The cost of offering live and on demand streaming for a content provider, or ISP, can be alleviated through

the use of a P2P system that offloads the streaming servers. This can give rise to new business models for subscription-based streaming services, in which viewers are given monetary incentives to upload and temporarily store video content.

Bibliography

[1] Host extensions for IP Multicasting - RFC 1112, 1989.

[2] IP Multicast Applications: Challenges and Solutions - RFC 3170, 2001.

[3] Akamai, http://www.akamai.com/html/technology/products/streaming.html.

[4] AT&T Content Delivery, http://www.business.att.com/enterprise/family/content-delivery/broadcast-video/.

[5] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the Internet. In *Proc. of ACM IMC*, 2004.

[6] T. Silverston and O. Fourmaux. Measuring P2P IPTV systems. In *Proc. of ACM NOSSDAV*, 2007.

[7] X. Hei, C. Liang, Y. Liu, and K.W. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9:1672–1687, December 2007.

[8] D. Ciullo, M. A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia. Network awareness of P2P live streaming applications: a measurement study. *IEEE Transactions on Multimedia*, 12:54–63, January 2010.

[9] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. of ACM SIGCOMM*, pages 205–217, 2002.

[10] Yang hua Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. A case for end system multicast. *IEEE Journal On Selected Areas in Communications*, 20(8), 2003.

[11] J. G. Apostolopoulos and M. D. Trott. Path diversity for enhanced media streaming. *IEEE Communications Magazine*, 42:80–87, 2004.

[12] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proc. of ACM NOSSDAV*, pages 177–186, 2002.

[13] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in a cooperative environment. In *Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[14] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proc of ACM SIGCOMM*, 2004.

[15] T. Qiu, I. Nikolaidis, and F. Li. On the design of incentive-aware P2P streaming. *Journal of Internet Engineering*, 1:61–71, October 2007.

[16] M. Zhang, L. Zhao, Y. Tang, J.-G. Luo, and S.-Q. Yang. Large-scale live media streaming over peer-to-peer networks through global internet. In *Proc. of ACM workshop on Advances in peer-to-peer multimedia streaming (P2PMMS)*, pages 21–28, 2005.

[17] V. Venkataraman, K. Yoshida, and P. Francis. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proc. of the 14th IEEE International Conference on Network Protocols (ICNP)*, pages 2 –11, Nov. 2006.

[18] Bo Li, Susu Xie, Yang Qu, G.Y. Keung, Chuang Lin, Jiangchuan Liu, and Xinyan Zhang. Inside the new coolstreaming: Principles, measurements and performance implications. In *Proc. of IEEE INFOCOM*, pages 1031–1039, April 2008.

[19] E. Setton, J. Noh, and B. Girod. Congestion-distortion optimized peer-to-peer video streaming. In *Proc. of IEEE International Conference on Image Processing*, 2006.

[20] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17:1103–1120, 2007.

[21] B. Girod, K. Stuhlmüller, M. Link, and U. Horn. Packet loss resilient internet video streaming. In *Proc. of IEEE International Conference on Image Processing*, 2000.

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

[23] Y. Chu, J. Chuang, and H. Zhang. A case for taxation in peer-to-peer streaming broadcast. In *Proc. of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, 2004.

[24] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *SIAM Journal of Applied Mathematics*, 8:300–304, 1960.

[25] Gy. Dán, V. Fodor, and G. Karlsson. On the asymptotic behavior of end-point-based multimedia streaming. In *Proc. of International Zurich seminar on Communication*, 2006.

[26] Gy. Dán, V. Fodor, and G. Karlsson. On the stability of end-point-based multimedia streaming. In *Proc. of IFIP Networking*, 2006.

[27] Y. Liu. On the minimum delay peer-to-peer video streaming: how realtime can it be? In *Proc. of ACM Multimedia*, 2007.

[28] L. Massoulie et al. Randomized decentralized broadcasting algorithms. In *Proc. of IEEE INFOCOM*, 2007.

[29] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: Optimal performance trade-offs. In *Proc. of ACM SIGMETRICS*, 2008.

[30] Luca Abeni, Csaba Kiraly, and Renato Lo Cigno. On the optimal scheduling of streaming applications in unstructured meshes. In *Proc. of IFIP/TC6 Networking*, pages 117–130, 2009.

[31] F. Picconi and L. Massoulié. Is there a future for mesh-based live video streaming? In *IEEE International Conference on Peer-to-Peer Computing*, 2008.

[32] C. Liang, Y. Guo, and Y. Liu. Is random scheduling sufficient in P2P video streaming? In *Proc. of the 28th International Conference on Distributed Computing Systems (ICDCS)*, 2008.

[33] Y. Guo, C. Liang, and Y. Liu. Adaptive queue-based chunk scheduling for P2P live streaming. In *Proc. of IFIP Networking*, 2008.

[34] C. Kiraly, L. Abeni, and R. Lo Cigno. Effects of P2P streaming on video quality. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 1 –5, May 2010.

[35] E. Setton, J. Noh, and B. Girod. Low latency video streaming over peer-to-peer networks. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, pages 569 –572, july 2006.

[36] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs. In *Proc. of the 10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1 –10, August 2010.

[37] STUN - Simple traversal of user datagram protocol (UDP) through network address translators (NATs) - IETF RFC 3489, March 2003.

[38] S. Guha and P. Francis. Characterization and measurement of TCP traversal through NATs and firewalls. In *Proc. of the 5th ACM SIGCOMM conference on Internet Measurement (IMC)*, 2005.

[39] T. Small, B. Liang, and B. Li. Scaling laws and tradeoffs in peer-to-peer live multimedia streaming. In *Proc. of ACM Multimedia*, 2006.

[40] M. Bishop, S. Rao, and K. Sripanidkulchai. Considering priority in overlay multicast protocols under heterogeneous environments. In *Proc. of IEEE INFOCOM*, 2006.

[41] Y. Sung, M. Bishop, and S. Rao. Enabling contribution awareness in an overlay broadcasting system. In *Proc. of ACM SIGCOMM*, pages 411–422, 2006.

[42] R. J. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo. Adaptive overlay topology for mesh-based P2P-TV systems. In *Proc. of ACM NOSSDAV*, 2009.

[43] D.-N. Ren, Y.-T. H. Li, and S.-H. Chan. On reducing mesh delay for peer-to-peer live streaming. In *Proc. of IEEE INFOCOM*, 2008.

[44] Y. Cui and K. Nahrstedt. Layered peer-to-peer streaming. In *Proc. of ACM NOSSDAV*, 2003.

[45] Pierpaolo Baccichet, Thomas Schierl, Thomas Wiegand, and Bernd Girod. Low-delay peer-to-peer streaming using scalable video coding. In *Packet Video 2007*, pages 173 –181, Nov. 2007.

[46] K. Park, S. Pack, and T. Kwon. Climber: An incentive-based resilient peer-to-peer system for live streaming services. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[47] A. Habib and J. Chuang. Service differentiated peer selection: An incentive mechanism for peer-to-peer media streaming. *IEEE Transactions on Multimedia*, 8:610–621, June 2009.

[48] G. Tan and S. A. Jarvis. A payment-based incentive and service differentiation mechanism for peer-to-peer streaming broadcast. In *Proc. of 14th International Workshop on Quality of Service (IWQoS)*, 2006.

[49] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang. Using layered video to provide incentives in P2P live streaming. In *Proc. of Workshop on Peer-to-Peer Streaming and IP-TV*, 2007.

[50] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang. P2P video live streaming with MDC: Providing incentives for redistribution. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, 2007.

[51] T. Chahed, E. Altman, and S.E. Elayoubi. Joint uplink and downlink admission control to both streaming and elastic flows in CDMA/HSDPA systems. *Performance Evaluation*, 65:869–882, November 2008.

[52] Gy. Dán and V. Fodor. Delay asymptotics and scalability for peer-to-peer live streaming. *IEEE Transactions on Parallel and Distributed Systems*, 20(10):1499–1511, October 2009.

[53] Bo Li, G.Y. Keung, Susu Xie, Fangming Liu, Ye Sun, and Hao Yin. An empirical study of flash crowd dynamics in a P2P-based live video streaming system. In *Proc. of IEEE GLOBECOM*, pages 1–5, December 2008.

[54] I. Bermudez, M. Mellia, and M. Meo. Passive characterization of SopCast usage in residential ISPs. In *Proc. of 11th IEEE International Conference on Peer-to-Peer Computing*, 2011.

[55] Fangming Liu, Bo Li, Lili Zhong, Baochun Li, and Di Niu. How P2P streaming systems scale over time under a flash crowd? In *Proc. of the 8th International Conference on Peer-to-Peer Systems (IPTPS)*, 2009.

[56] Zhijia Chen, Bo Li, G. Keung, Hao Yin, Chuang Lin, and Yuanzhuo Wang. How scalable could P2P live media streaming system be with the stringent time constraint? In *Proc. of IEEE International Conference on Communications (ICC)*, pages 1 –5, June 2009.

[57] E. Biersack, W. Geyer, and C. Bernhardt. Intra- and inter-stream synchronisation for stored multimedia streams. In *Proc. of the Third IEEE International Conference on Multimedia Computing and Systems*, pages 372 –381, jun 1996.

[58] P. Goyal, H.M. Vin, C. Shen, and P.J. Shenoy. A reliable, adaptive network protocol for video transport. In *IEEE INFOCOM*, mar 1996.

[59] Donald L. Stone and Kevin Jeffay. An empirical study of a jitter management scheme for video teleconferencing. *Multimedia Systems*, 2(6):267–279, 1995.

[60] N. Laoutaris and I. Stavrakakis. Adaptive playout strategies for packet video receivers with finite buffer capacity. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 969 –973, 2001.

[61] M. Kalman, E. Steinbach, and B. Girod. Adaptive media playout for low-delay video streaming over error-prone channels. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(6):841–851, june 2004.

[62] Y.F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang. Modeling the impact of frame rate on perceptual quality of video. In *Proc. of 15th IEEE International Conference on Image Processing (ICIP)*, pages 689–692, 2008.

[63] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *Proc. of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 6–6, 2005.

[64] http://www.dslreports.com/shownews/new-comcast-throttling-system-100-online-100015.

[65] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can isps and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37:29–40, July 2007.

[66] Gy. Dán, T. Hossfeld, S. Oechsner, P. Cholda, R. Stankiewicz, I. Papafili, and G.D. Stamoulis. Interaction patterns between P2P content distribution systems and isps. *IEEE Communications Magazine*, 49(5):222 –230, May 2011.

[67] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4P: Provider portal for applications. In *Proc. of the ACM SIGCOMM 2008*, pages 351–362, 2008.

[68] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proc. of the International Conference on Distributed Computer Systems (ICDCS)*, 2006.

[69] IETF ALTO Working Group, https://datatracker.ietf.org/wg/alto/charter/.

[70] S. Le Blond, A. Legout, and W. Dabbous. Pushing bittorrent locality to the limit. *Computer Networks*, 2010.

[71] D. Choffnes and F. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *Proc. of ACM SIGCOMM*, 2008.

[72] F. Picconi and L. Massoulié. ISP-friend or foe? making P2P live streaming ISP-aware. In *Proc. of the 29thIEEE International Conference on Distributed Computer Systems (ICDCS)*, pages 413–422, 2009.

[73] Xin Jin and Yu-Kwong Kwok. Network aware P2P multimedia streaming: Capacity or locality? In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 54 –63, Sept. 2011.

[74] J. Seedorf, S. Niccolini, M. Stiemerling, E. Ferranti, and R. Winter. Quantifying operational cost-savings through alto-guidance for P2P live streaming. In *3rd Workshop on Economic Traffic Management (ETM 2010)*, September 2010.

[75] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proc.of ACM SIGCOMM*, pages 75–86, 2010.

[76] A. Dhamdhere and C. Dovrolis. The Internet is flat: modeling the transition from a transit hierarchy to a peering mesh. In *Proc. of ACM Co-NEXT*, pages 1–12, 2010.

Paper A

Streaming Performance in Multiple-tree-based Overlays

György Dán, Viktória Fodor and Ilias Chatzidrossos.
In Proceedings of IFIP Networking, Atlanta, Georgia, May 2007.

Streaming Performance in Multiple-tree-based Overlays

György Dán, Viktória Fodor and Ilias Chatzidrossos
Laboratory for Communication Networks
School of Electrical Engineering
KTH, Royal Institute of Technology
E-mail: {gyuri,vfodor,iliasc}@ee.kth.se

Abstract

In this paper we evaluate the data transmission performance of a generalized multiple-tree-based overlay architecture for peer-to-peer live streaming that employs multipath transmission and forward error correction. We give mathematical models to describe the error recovery in the presence of packet losses. We evaluate the data distribution performance of the overlay, its asymptotic behavior, the stability regions for the data transmission, and analyze the system behavior around the stability threshold. We argue that the composed measure of the mean and the variance of the packet possession probability can support adaptive forward error correction.

1 Introduction

The success of peer-to-peer overlays for live multicast streaming depends on their ability to maintain acceptable perceived quality at all the peers, that is, to provide data transmission with low delay and information loss. Live streaming solutions usually apply multiple distribution trees and some form of error control to deal with packet losses due to congestion and peer departures. In these systems peers have to relay data to their child nodes with low delay, which limits the possibilities of error recovery. Consequently, the main problem to be dealt with is the spatial propagation and thus the accumulation of losses, which results in low perceived quality for peers far from the source.

Several works deal with the management of the overlay, with giving incentives for collaboration, with peer selection and tree reconstruction considering peer het-

[1]This work was in part supported by the Swedish Foundation for Strategic Research through the projects Winternet and AWSI, and by Wireless@KTH

erogeneity and the underlying network topology ([1, 2, 3, 4, 5, 6] and references therein).

In [7] the authors propose time shifting and video patching to deal with losses and discuss related channel allocation and group management issues. In [8] robustness is achieved by distributing packets to randomly chosen neighbors outside the distribution tree. In [9] retransmission of the lost data is proposed to limit temporal error propagation. CoopNet [10] and SplitStream [11] propose the use of multiple distribution trees and a form of multiple description coding (MDC) based on forward error correction (FEC). In the case of packet losses peers can stop error propagation by reconstructing the video stream from the set of received substreams using error correcting codes.

There are several designs proposed and also implemented, the evaluation of these solutions is however mostly based on simulations and measurements. Our goal is to define abstract models of peer-to-peer streaming overlays that help us to understand some basic characteristics of streaming in multiple transmission trees and thus, can support future system design.

Previously, we proposed mathematical models to describe the behavior of CoopNet like architectures in [12, 13, 14]. Our results showed that the two architectures proposed in the literature, and used as a basis in recent works (e.g., [2, 15]) are straightforward but not optimal. Minimum depth trees minimize the number of affected peers at peer departure, minimize the effect of error propagation from peer to peer, and introduce low transmission delay. Nevertheless, the overlay is unstable and may become disconnected when one of the trees runs out of available capacity after consecutive peer departures. Minimum breadth trees are stable and easy to manage, but result in long transmission paths. Thus many nodes are affected if a peer leaves, there may be large delays, and the effect of error propagation may be significant.

In [16] we proposed a generalized multiple-tree-based architecture and showed that the stability of the overlay can be increased significantly by the flexible allocation of peer output bandwidth across several transmission trees. In this paper, we evaluate the performance of this generalized architecture. First we show how the allocation of peer output bandwidth affects the data distribution performance and discuss whether proper bandwidth allocation can lead to both increased overlay stability and good data distribution performance. We show that the packet possession probability at the peers decreases ungracefully if the redundancy level is not adequate, and evaluate how the variance of the packet possession probability can predict quality degradation.

The rest of the paper is organized as follows. Section 2 describes the considered overlay structure and error correction scheme. We evaluate the stability of the data distribution in Section 3. Section 4 discusses the performance of the overlay based on the mathematical models and simulations and we conclude our work in Section 5.

2 System description

The peer-to-peer streaming system discussed in this paper is based on two key ideas: the overlay consists of multiple transmission trees to provide multiple transmission paths from the sender to all the peers, and data transmission is protected by block based FEC.

2.1 Overlay structure

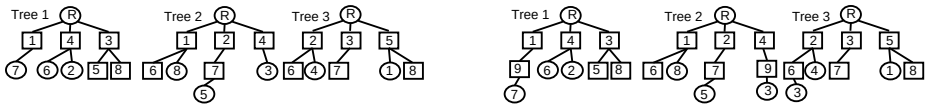
The overlay consists of the streaming server and N peer nodes. The peer nodes are organized in t distribution trees with the streaming server as the root of the trees. The peers are members of all t trees, and in each tree they have a different parent node from which they receive data. We denote the maximum number of children of the root node in each tree by m, and we call it the multiplicity of the root node. We assume that nodes do not contribute more bandwidth towards their children as they use to download from their parents, which means, that each node can have up to t children to which it forwards data.

A node can have children in up to d of the t trees, called the fertile trees of the node. A node is sterile in all other trees, that is, it does not have any children. We discuss two different policies that can be followed to allocate output bandwidth in the fertile trees. With the *unconstrained* capacity allocation (UCA) policy a node can have up to t children in any of its fertile trees. With the *balanced* capacity allocation (BCA) policy a node can have up to $\lceil t/d \rceil$ children in any of its fertile trees.

By setting $d = t$ one gets the minimum breadth tree described in [10], and by setting $d = 1$ one gets the minimum depth tree evaluated in [2, 6, 10, 15]. For $1 < d < t$ the number of layers in the overlay is $O(logN)$ as for $d = 1$.

The construction and the maintenance of the trees can be done either by a distributed protocol (structured, like in [11] or unstructured, like in [9]) or by a central entity, like in [10]. The results presented in this paper are not dependent on the particular solution used. Nevertheless, we defined a centralized overlay construction algorithm in [16]. The objective of the algorithm is to minimize the depth of the trees and the probability that trees run out of free capacity after node departures. This is achieved by pushing sterile nodes to the lower layers of the trees and by assigning arriving nodes to be fertile in trees with the least available capacity. If we denote the number of layers in the trees by L, then in a well maintained tree each node is $1 \leq i \leq L$ hops away from the root node in its fertile trees, and $L - 1 \leq i \leq L$ hops away in its sterile trees. As shown in [16], increasing the number of fertile trees increases the overlay stability, with significant gain already at $d = 2$, and the UCA policy giving the highest increase.

Fig. 1 shows an overlay constructed with the proposed algorithm for $t = 3$, $m = 3$ and $d = 2$, also showing how the overlay changes when a new node joins.



a) b)

Figure 1: a) Overlay with $N = 8, t = 3, m = 3$ and $d = 2$, b) the same overlay with $N = 9$. Identification numbers imply the order of arrival, squares indicate that the node is fertile.

2.2 Data distribution with forward error correction

The root uses block based FEC, e.g., Reed-Solomon codes [17], so that nodes can recover from packet losses due to network congestion and node departures. To every k packets of information it adds c packets of redundant information, which results in a block length of $n = k + c$. We denote this FEC scheme by FEC(n,k). If the root would like to increase the ratio of redundancy while maintaining its bitrate unchanged, then it has to decrease its source rate. Lost packets can be reconstructed as long as at most c packets are lost out of n packets. The root sends every t^{th} packet to its children in a given tree in a round-robin manner. If $n \leq t$ then at most one packet of a block is distributed over the same distribution tree. Peer nodes relay the packets upon reception to their respective child nodes. Once a node receives at least k packets of a block of n packets it recovers the remaining c packets and forwards the ones belonging to its fertile trees.

3 Data distribution model

We use two metrics to measure the performance of the data distribution in the overlay. The first metric is the probability π that an arbitrary node receives or can reconstruct (i.e., possesses) an arbitrary packet. If we denote by ρ_r the number of packets possessed by node r in an arbitrary block of packets, then π can be expressed as the average ratio of packets possessed in a block over all nodes, i.e., $\pi = E[\sum_r \rho_r/n/N]$. The second metric is σ, the average standard deviation of ρ_r/n over all nodes, i.e., $\sigma^2 = E[\sum_r (\rho_r/n)^2/N] - \pi^2$.

The mathematical model we present describes the behavior of the overlay in the presence of independent packet losses and without node dynamics. We denote the probability that a packet is lost between two adjacent nodes by p. We assume that the probability that a node is in possession of a packet is independent of that a node in the same layer is in possession of a packet. We also assume that nodes can wait for redundant copies to reconstruct a packet for an arbitrary amount of time. For the model we consider a tree with the maximum number of nodes in the last layer, hence nodes are fertile in layers $1..L-1$ and are sterile in layer L. For simplicity, we assume that $n = t$ and t/d is an integer. We will comment on the possible effects of our assumptions later. The model assumes the BCA policy. By

modifying the weights in (1) the model can be used to consider other policies as well. For brevity we restrict the analytical evaluation to this case, and compare the results to simulations with other policies.

Hence, our goal is to calculate $\overline{\pi}^{(z)}= E[\sum_r (\rho_r/n)^{(z)}/N] = \sum_r E[(\rho_r/n)^{(z)}]/N$, the average of the z^{th} moment ($z \in \{1,2\}$) of the ratio of possessed packets. We introduce $\pi(i)^{(z)}$, the z^{th} moment of the ratio of possessed packets for a node that is in layer i in its fertile trees. For simplicity we assume that nodes are in the same layer in their fertile trees. We can express $\overline{\pi}^{(z)}$ by weighting the $\pi(i)^{(z)}$ with the portion of nodes that are in layer i of their fertile trees.

$$\overline{\pi}^{(z)} = \sum_{i=1}^{L-1} \frac{(t/d)^{i-1}}{((t/d)^{L-1}-1)/(t/d-1)} \pi(i)^{(z)}. \tag{1}$$

To calculate $\pi(i)^{(z)}$ we have to calculate the probabilities $\pi_f(i)$ that a node, which is in layer i in its fertile tree, receives or can reconstruct an *arbitrary packet in its fertile tree*. Since the root node possesses every packet, we have that $\pi_f(0) = 1$. The probability that a node in layer i receives a packet in a tree is $\pi_a(i) = (1-p)\pi_f(i-1)$. A node can possess a packet in its fertile tree either if it receives the packet or if it can reconstruct it using the packets received in the other trees. Reconstruction can take place if the number of received packets is at least k out of the remaining $n-1$, hence we can write for $1 \le i \le L-1$

$$\pi_f(i) = \pi_a(i) + \left\{ (1-\pi_a(i)) \sum_{j=k}^{n-1} \sum_{u=max(0,j-n+d)}^{min(j,d-1)} \binom{d-1}{u} \pi_a(i)^u (1-\pi_a(i))^{d-1-u} \right.$$
$$\left. \binom{n-d}{j-u} \pi_a(L)^{j-u} (1-\pi_a(L))^{n-d-j+u} \right\}. \tag{2}$$

Based on the probabilities $\pi_f(i)$ we can express $\pi(i)^{(z)}$ ($1 \le i \le L-1$). If a node receives at least k packets in a block of n packets then it can use FEC to reconstruct the lost packets, and hence possesses all n packets. Otherwise, FEC cannot be used to reconstruct the lost packets. Packets can be received in the d fertile trees and in the $t-d$ sterile trees. Hence for $\pi(i)^{(z)}$ we get the equation

$$\pi(i)^{(z)} = \frac{1}{n} \sum_{j=1}^{n-d} \sum_{u=1}^{d} \tau(j+u)^z \binom{d}{u} \pi_a(i)^u (1-\pi_a(i))^{d-u} \tag{3}$$
$$\binom{n-d}{j} \pi_a(L)^j (1-\pi_a(L))^{n-d-j},$$

where $\tau(j)$ indicates the number of packets possessed after FEC reconstruction if j packets have been received:

$$\tau(j) = \begin{cases} j & 0 \le j < k \\ n & k \le j \le n. \end{cases}$$

We use an iterative method to calculate the probabilities $\pi_f(i)$. First, we set $\pi_f(L-1)^{(0)} = (1-p)^{L-1}$ and calculate the probabilities $\pi_f(i)^{(1)}$, $1 \leq i < L$. Then, in iteration r, we calculate $\pi_f(i)^{(r)}$, $1 \leq i < L$ using $\pi_f(L-1)^{(r-1)}$. The iteration stops when $|\pi_f(L-1)^{(r-1)} - \pi_f(L-1)^{(r)}| < \epsilon$, where $\epsilon > 0$. The $\pi(i)^{(z)}$ can then be calculated using (3). The calculation of π and σ are straightforward, since $\pi = \overline{\pi}^{(1)}$, and $\sigma^2 = \overline{\pi}^{(2)} - \pi^2$

3.1 Asymptotic behavior for large N

In the following we give an asymptotic bound on π to better understand its evolution. It is clear that $\pi_f(i)$ is a non-increasing function of i and $\pi_f(i) \geq 0$. Hence, we can give an upper estimate of $\pi_f(i)$ by assuming that the nodes that forward data in layer i are sterile in the same layer. Then, instead of (2) we get the following nonlinear recurrence equation

$$\begin{aligned} \overline{\pi}_f(i+1) &= \overline{\pi}_a(i+1) + \\ &\quad (1-\overline{\pi}_a(i+1)) \sum_{j=n-c}^{n-1} \left(\begin{array}{c} n-1 \\ j \end{array} \right) \overline{\pi}_a(i+1)^j (1-\overline{\pi}_a(i+1))^{n-1-j}. \end{aligned} \tag{4}$$

This equation is the same as (2) in [13], and thus the analysis shown there can be applied to describe the evolution of $\overline{\pi}_f(i)$. For brevity, we only state the main results regarding $\overline{\pi}_f(i)$, for a detailed explanation see [13]. For every (n,k) there is a loss probability p_{max} below which the packet possession probability $\overline{\pi}_f(\infty) > 0$ and above which $\overline{\pi}_f(\infty) = 0$. Furthermore, for any $0 < \delta < 1$ there is (n,k) such that $\overline{\pi}_f(\infty) \geq \delta$.

Consequently, in the considered overlay if $p > p_{max}$, then $\lim_{N\to\infty} \pi = 0$, because $\overline{\pi}_f(i+1) \geq \pi_f(i+1) \geq \pi(i+1)^{(1)}$, and $\lim_{N\to\infty} \overline{\pi}_f(L-1) = 0$. For $p < p_{max}$ stability depends on the number of layers in the overlay and the FEC block length due to the initial condition $\pi_f(L-1)^{(0)} = (1-p)^{L-1}$, but not directly on the number of nodes. This explains why placing nodes with large outgoing bandwidths close to the root improves the overlay's performance [2, 6]. In the case of stability $\pi_f(i) \geq \overline{\pi}_f(\infty) > 0$ and $\pi(i)^{(1)} \geq \overline{\pi}_f(\infty) > 0$.

3.2 Discussion of the assumptions

In the following we discuss the validity of certain assumptions made in the model. The model does not take into account the correlations between packet losses in the Internet. Nevertheless, it can be extended to heterogeneous and correlated losses by following the procedure presented in [13] for the minimum breadth trees. Losses occurring in bursts on the output links of the nodes influence the performance of the overlay if several packets of the same block are distributed over the same tree, that is, if $n > t$. Bursty losses in the backbone influence the performance if packets of different distribution trees traverse the same bottleneck. The analysis in [13]

showed that loss correlations on the input links and heterogeneous losses slightly decrease the performance of the overlay.

The model can be extended to nodes with heterogeneous input and output bandwidths. The procedure is similar to that followed when modeling heterogeneous losses [13], but the effects of the heterogeneous bandwidths on the trees' structure have to be taken into account. We decided to show equations for the homogeneous case here to ease understanding.

In the analysis we assume that the number of nodes in the last layer of the tree is maximal. If the number of nodes in the last layer of the tree is not maximal then some nodes are in layer $L-1$ in their sterile trees, and the overlay's performance becomes slightly better. The results of the asymptotic analysis still hold however.

Our results for block based FEC apply to PET and the MDC scheme considered in [10], where different blocks (layers) of data are protected with different FEC codes. The packet possession probability for the different layers depends on the strength of the FEC codes protecting them, and can be calculated using the model.

We do not model the temporal evolution of the packet possession probability. We use simulations to show that the performance predicted by the model is achieved within a time suitable for streaming applications.

The model does not take into account node departures, an important source of disturbances for the considered overlay. Following the arguments presented in [13] node departures can be incorporated in the model as an increase of the loss probability by $p_\omega = N_d/N \times \theta$, where N_d is the mean number of departing nodes per time unit and θ is the time nodes need to recover (discovery and reconnection) from the departure of a parent node. The simulation results presented in [13] show that for low values of p_ω this approximation is accurate.

The results of the model apply for $n < t$ without modifications, and a similar model can be developed for $n > t$ by considering the distribution of the number of packets possessed by the nodes in their fertile trees. However, for $n > t$ node departures lead to correlated losses in the blocks of packets, which aggravates their effect on the performance.

4 Performance evaluation

In the following we analyze the behavior of the overlay using the analytical model presented in the previous section and via simulations. For the simulations we developed a packet level event-driven simulator.

We used the GT-ITM [18] topology generator to generate a transit-stub model with 10000 nodes and average node degree 6.2. We placed each node of the overlay at random at one of the 10000 nodes and used the one way delays given by the generator between the nodes. The mean delay between nodes of the topology is 44 ms. The delay between overlay nodes residing on the same node of the topology was set to 1 ms. Losses on the paths between the nodes of the overlay occur

independent of each other according to a Bernoulli model with loss probability p. We consider the streaming of a 112.8 kbps data stream to nodes with link capacity 128 kbps. The packet size is 1410 bytes. Nodes have a playout buffer capable of holding 140 packets, which corresponds to 14 s of playout delay. Each node has an output buffer of 80 packets to absorb the bursts of packets in its fertile trees.

We assume that session holding times follow a log-normal distribution with mean $1/\mu = 306s$ and that nodes join the overlay according to a Poisson process with $\lambda = \overline{N}\mu$, supported by studies [19, 20]. To obtain the results for a given overlay size $\overline{N}$, we start the simulation with $\overline{N}$ nodes in its steady state as described in [21] and let nodes join and leave the overlay for 5000 s. The measurements are made after this warm-up period for a static tree over 1000 s and the presented results are the averages of 10 simulation runs. The results have less than 5 percent margin of error at a 95 percent level of confidence.

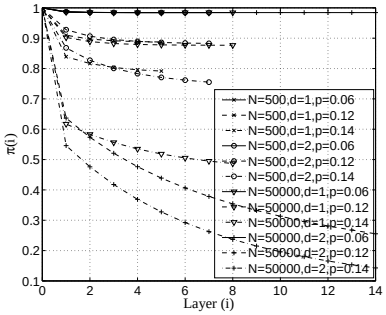


Figure 2: $\pi(i)$ vs. i for $m = t = n = 4$, $c = 1$.

Fig. 2 shows $\pi(i)$ as a function of i for $t = n = 4$, $c = 1$, and different overlay sizes and values of d as obtained by the mathematical model (i.e., the BCA policy). The value of the threshold for the FEC(4,3) code is $p_{max} = 0.129$. The figure shows that when the overlay is stable (e.g., $\forall d, N$ at $p = 0.06$), neither its size nor d has a significant effect on $\pi(i)$. However, in the unstable state ($d = 2, N = 50000$ at $p = 0.12$; $\forall d, N$ at $p = 0.14$) both increasing N and increasing d decrease $\pi(i)$, as the number of layers in the overlay increases in both cases.

Fig. 3 shows π as a function of p obtained with the mathematical model for $m = 4$ and $n = t$. The vertical bars show the values $\pi(1)$ at the upper end and $\pi(L-1)$ at the lower end. We included them for $d = 1$ only to ease readability, but they show the same properties for other values of d as well. The figure shows that π remains high and is unaffected by N and d as long as the overlay is stable. It drops however once the overlay becomes unstable, and the drop of the packet possession probability gets worse as the number of nodes and hence the number of layers in the overlay increases. At the same time the difference between $\pi(1)$ and $\pi(L-1)$ (the packet possession probability of nodes that are fertile in the first and the penultimate layers, respectively) increases. Furthermore, increasing t (and hence n) increases π in a stable system, but the stability region gets smaller and the drop of the packet possession probability gets faster in the unstable state due to the longer FEC codes. The curves corresponding to $\overline{\pi}_f(\infty)$ show the value of the asymptotic bound calculated using (4).

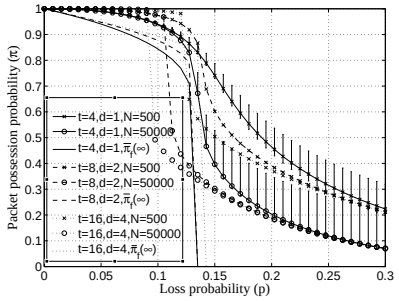


Figure 3: π vs. p for $m = 4$, $n = t$, $k/n = 0.75$.

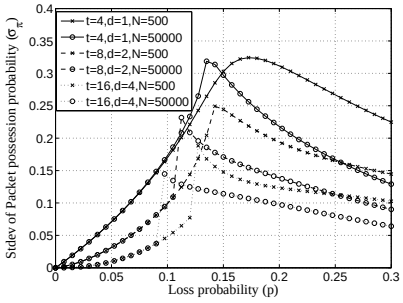


Figure 4: σ_π vs. p for $m = 4$, $n = t$, $k/n = 0.75$.

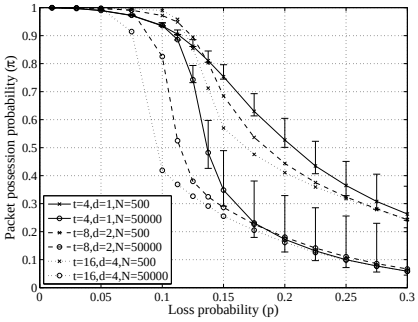


Figure 5: π vs. p for $m = 4$, $n = t$, $k/n = 0.75$. BCA policy, simulation results.

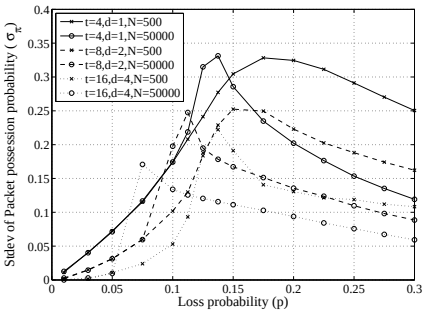


Figure 6: σ_π vs. p for $m = 4$, $n = t$, $k/n = 0.75$. BCA policy, simulation results.

Due to the ungraceful degradation of π it is difficult to maintain the stability of the overlay in a dynamic environment by measuring the value of π. Hence, we look at the standard deviation of the packet possession probability. Fig. 4 shows the standard deviation σ_π as a function of p obtained with the mathematical model for $m = 4$ and $n = t$. The standard deviation increases rapidly even for low values of p and reaches its maximum close to p_{max}. Increasing the value of t decreases the standard deviation of π, i.e., the number of packets received in a block varies less among the nodes of the overlay. Its quick response to the increase of p makes σ_π more suitable for adaptive error control than π itself.

To validate our model we first present simulation results for the BCA policy. Figs. 5 and 6 show π and σ_π respectively as a function of p for the same scenarios as Figs. 3 and 4. Both figures show a good match with the analytical model and confirm that the increase of the standard deviation is a good indicator of the increasing

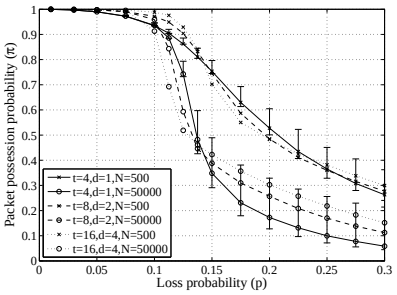


Figure 7: π vs. p for $m = 4$, $n = t$, $k/n = 0.75$. UCA policy, simulation results.

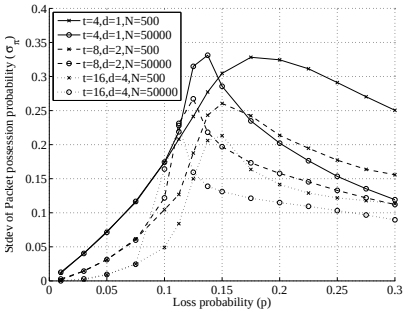


Figure 8: σ_π vs. p for $m = 4$, $n = t$, $k/n = 0.75$. UCA policy, simulation results.

packet loss probability. For long FEC codes the simulation results show slightly worse performance close to the stability threshold compared to the analytical results. The difference is due to late arriving packets, i.e., FEC reconstruction is not possible within the time limit set by the playout buffer's size.

To see the effects of the capacity allocation policy, we show π as a function of p in Fig. 7 for the same scenarios as in Fig. 5 but for the UCA policy. Comparing the figures we see that π is the same in the stable state, but is higher in the unstable state of the overlay. Furthermore, the region of stability is wider. Comparing the results for σ_π shown in Fig. 8 we can observe that for $\overline{N} = 50000$ and $d > 1$ the standard deviation is somewhat higher compared to the BCA policy and is similar in shape to the $d = 1$ case. This and the wider region of stability using the UCA policy is due to that the overlay has less layers than using the BCA policy as it is shown in Fig. 9. The figure shows the cumulative distribution function (CDF) of the layer where the nodes of the overlay are in their sterile trees.

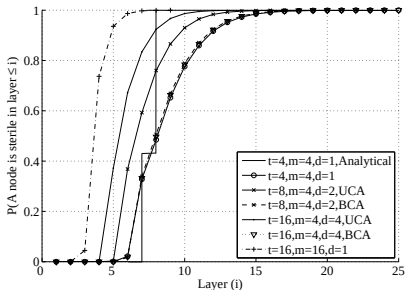


Figure 9: CDF of the layer where nodes are sterile for $N = 50000$. Simulation results.

There is practically no difference between the distributions for $d = 1$ and the BCA policy with $d > 1$ for the same t/d value. With the UCA policy nodes tend to have more children in the fertile tree where they are closest to the root due to the parent selection algorithm, so that the trees' structure is similar to the $d = 1$ case

and the number of layers is lower than using BCA (compare the results for $t = 16$, $d = 1$ vs. $t = 16$, $d = 4$, UCA vs. $t = 16$, $d = 4$, BCA).
Hence, the data distribution performance of an overlay with t trees and $d = 1$ can be closely resembled with an overlay with $d > 1$ and td trees by employing a (centralized or distributed) mechanism that promotes parents close to the root, such as the UCA policy. Doing so allows the use of longer FEC codes, hence better performance in the stable region but a similar stability region due to the lower number of layers. At the same time one can decrease the probability that a node fails to reconnect to the overlay after the departure of a parent node [16]. Longer FEC codes decrease the variance of the packet possession probability and allow a smoother adaptation of the redundancy as a function of the measured network state.

5 Conclusion

In this paper, we analyzed a peer-to-peer live streaming solution based on multiple transmission trees, FEC and free allocation of the output bandwidth of the peers across several trees. The aim of this design is to avoid tree disconnections after node departures, which can happen with high probability in resource scarce overlays if all the peers can forward data in one tree only.

We presented a mathematical model to express the packet possession probability in the overlay for the case of independent losses and the BCA policy. We determined the stability regions as a function of the loss probability between the peers, of the number of layers and of the FEC block length, and analyzed the asymptotic behavior of the overlay for a large number of nodes. We calculated the variance of the packet possession probability to study the overlay around the stability threshold. We concluded that the variance increases significantly with the packet loss probability between the peers and consequently it is a good performance measure for adaptive forward error correction. It will be subject of future work to design a robust stabilizing controller that can maintain a target packet possession probability in a dynamic environment.

We concluded that as long as the overlay is stable, the performance of the data transmission is not influenced by the number of fertile trees and the allocation policy, while longer FEC codes improve it. Increasing the number of fertile trees decreases however the packet possession probability in the overlay in the unstable region due to longer transmission paths. Nevertheless, with the UCA policy one can increase the number of trees, that of the fertile trees and the FEC block length, while the performance can be close to that of the minimum depth trees, because the UCA policy leads to shallow tree structures. These results show that adjusting the number of fertile trees can be a means to improve the overlays' stability without deteriorating the performance of the data distribution.

References

[1] Liao, X., Jin, H., Liu, Y., Ni, L., Deng, D.: Anysee: Scalable live streaming service based on inter-overlay optimization. In: Proc. of IEEE INFOCOM. (April 2006)

[2] Bishop, M., Rao, S., Sripanidkulchai, K.: Considering priority in overlay multicast protocols under heterogeneous environments. In: Proc. of IEEE INFOCOM. (April 2006)

[3] Cui, Y., Nahrstedt, K.: Layered peer-to-peer streaming. In: Proc. of NOSSDAV. (2003) 162–171

[4] Cui, Y., Nahrstedt, K.: High-bandwidth routing in dynamic peer-to-peer streaming. In: Proc. of ACM APPMS. (2005) 79–88

[5] Tan, G., S.A., J.: A payment-based incentive and service differentiation mechanism for peer-to-peer streaming broadcast. In: Proc. of IEEE IWQoS. (2006) 41–50

[6] Sung, Y., Bishop, M., Rao, S.: Enabling contribution awareness in an overlay broadcasting system. In: Proc. of ACM SIGCOMM. (2006) 411–422

[7] Guo, M., Ammar, M.: Scalable live video streaming to cooperative clients using time shifting and video patching. In: Proc. of IEEE INFOCOM. (2004)

[8] Banerjee, S., Lee, S., Braud, R., Bhattacharjee, B., Srinivasan, A.: Scalable resilient media streaming. In: Proc. of NOSSDAV. (2004)

[9] Setton, E., Noh, J., Girod, B.: Rate-distortion optimized video peer-to-peer multicast streaming. In: Proc. of ACM APPMS. (2005) 39–48

[10] Padmanabhan, V., Wang, H., Chou, P.: Resilient peer-to-peer streaming. In: Proc. of IEEE ICNP. (2003) 16–27

[11] Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A., Singh, A.: SplitStream: High-bandwidth multicast in a cooperative environment. In: Proc. of ACM SOSP. (2003)

[12] Dán, G., Fodor, V., Karlsson, G.: On the asymptotic behavior of end-point-based multimedia streaming. In: Proc. of Internat. Zürich Seminar on Communication. (2006)

[13] Dán, G., Fodor, V., Karlsson, G.: On the stability of end-point-based multimedia streaming. In: Proc. of IFIP Networking. (May 2006) 678–690

[14] Dán, G., Chatzidrossos, I., Fodor, V., Karlsson, G.: On the performance of error-resilient end-point-based multicast streaming. In: Proc. of IWQoS. (June 2006) 160–168

[15] Sripanidkulchai, K., Ganjam, A., Maggs, B., Zhang, H.: The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In: Proc. of ACM SIGCOMM. (2004) 107–120

[16] Dán, G., Chatzidrossos, I., Fodor, V.: On the performance of multiple-tree-based peer-to-peer live streaming. In: Proc. of IEEE INFOCOM. (May 2007)

[17] Reed, I., Solomon, G.: Polynomial codes over certain finite fields. SIAM J. Appl. Math. **8**(2) (1960) 300–304

[18] Zegura, E.W., Calvert, K., Bhattacharjee, S.: How to model an internetwork. In: Proc. of IEEE INFOCOM. (March 1996) 594–602

[19] Veloso, E., Almeida, V., Meira, W., Bestavros, A., Jin, S.: A hierarchical characterization of a live streaming media workload. In: Proc. of ACM IMC. (2002) 117–130

[20] Sripanidkulchai, K., Maggs, B., Zhang, H.: An analysis of live streaming workloads on the Internet. In: Proc. of ACM IMC. (2004) 41–54

[21] Le Boudec, J.Y., Vojnovic, M.: Perfect simulation and stationarity of a class of mobility models. In: Proc. of IEEE INFOCOM. (March 2004)

Paper B

Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates

Ilias Chatzidrossos, György Dán and Viktória Fodor.
In P2P Networking and Applications, vol.3, num.1, March 2010.

Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates

Ilias Chatzidrossos, György Dán, Viktoria Fodor
KTH, Royal Institute of Technology,
School of Electrical Engineering
e-mail: {iliasc, gyuri, vfodor}@ee.kth.se

Abstract

In mesh-based peer-to-peer streaming systems data is distributed among the peers according to local scheduling decisions. The local decisions affect how packets get distributed in the mesh, the probability of duplicates and consequently, the probability of timely data delivery. In this paper we propose an analytic framework that allows the evaluation of scheduling algorithms. We consider four solutions in which scheduling is performed at the forwarding peer, based on the knowledge of the playout buffer content at the neighbors. We evaluate the effectiveness of the solutions in terms of the probability that a peer can play out a packet versus the playback delay, the sensitivity of the solutions to the accuracy of the knowledge of the neighbors' playout buffer contents, and the scalability of the solutions with respect to the size of the overlay. We also show how the model can be used to evaluate the effects of node arrivals and departures on the overlay's performance.

1 Introduction

Recent years have seen the proliferation of peer-to-peer (P2P) systems for the delivery of live streaming content on the Internet [1, 2, 3]. These systems are popular because they allow content providers to save on infrastructure costs: users contribute with their upload bandwidth resources to the distribution of the content. Whether P2P streaming systems will become an alternative to commercial, operator managed IPTV solutions depends on whether they can offer a comparable user experience.

The user experience in a streaming system is typically determined by two interdependent performance measures [4]: the playback delay and the playout probability. The playback delay determines how real-time streaming can be, but also

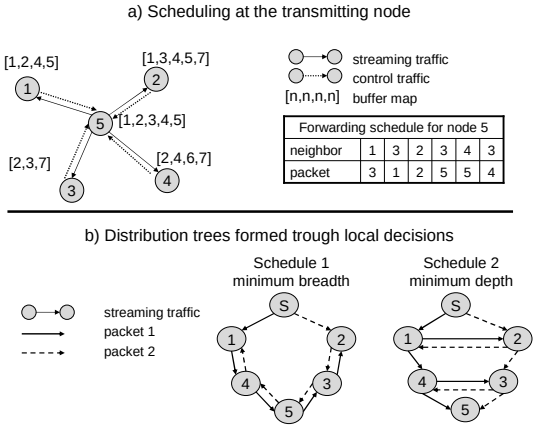


Figure 1: Mesh based streaming with local scheduling decision. a) Forwarding node 5 schedules neighbors and packets based on the received buffer maps. b) The distribution trees are formed through local decisions.

the zapping times, that is, the delays experienced when users switch channel. The playout probability is the ratio of packets successfully played out at the receiver. Additionally, the performance of a P2P streaming system can be evaluated with respect to two other metrics. First, the playback lag between the peers has to be low to provide similar real-time experience to all users. Second, the performance in terms of the previous three metrics has to scale well with the overlay's size.

P2P streaming systems proposed in the literature generally fall into one of two categories: multiple-tree based and mesh-based solutions [5]. Mesh-based solutions are considered to be more robust to dynamic overlay membership and fluctuating transmission capacities between the nodes, hence they are prevalent in commercial systems. Nevertheless as a consequence of their dynamic nature, there is a limited analytical understanding of the performance trade-offs of mesh-based solutions.

In a mesh-based system neighbor relations between the peers are determined locally, and consequently, there is no global structure maintained. The forwarding of data packets (or often a set of data packets) is decided locally at the peers, either on the transmitting or on the receiving side. As shown in Figure 1.a, the packet scheduling is based on information exchanged between the neighboring peers: peers report the packets they already possess, with the so called buffer map. Clearly, scheduling algorithms can be more efficient if they utilize detailed and timely information, and therefore the trade-off of streaming performance and control overhead has to be considered.

As a result of local scheduling decisions, the packets of the stream will reach the peers via different trees over the mesh. We call these trees the distribution trees.

The structure of the distribution trees is determined by the local decisions at the peers. Figure 1.b shows two examples where the distribution trees are minimum breadth and minimum depth trees respectively, with a depth of 5 and 3 overlay hops. The challenge of modelling mesh-based streaming comes then from the following closed loop: on one side the scheduling decisions determine the distribution trees formed over the mesh, and thus the availability of the packets at the peers, on the other side, the packet availability at the peers affects the scheduling decisions that can be made.

In this paper we propose an approach to cut this loop. We assume that the peers' positions in the distribution trees are statistically the same, and consequently, the probability that a peer receives a packet with given delay is the same for all peers. We validate our assumption with simulation and build then an analytic model to describe the packet reception process. We propose an analytic framework that can be used to evaluate the performance of different scheduling strategies. In this paper we consider four specific scheduling strategies, where scheduling is performed at the transmitting peers. We evaluate how the playout probability is affected by the number of neighbors, the playback delay and the frequency of buffer map updates. Then we evaluate the scalability of the solutions in terms of overlay size and derive conclusions on the structure of the distribution trees. We also use the model to evaluate the effects of node churn on the overlay performance.

The rest of the paper is organized as follows. In Section 2 we overview related work. Section 3 describes the P2P streaming systems we consider in this paper. We give the analytic model in Section 4, and evaluate the performance of the considered solutions in Section 5. Section 6 concludes our work.

2 Related Work

Several measurement studies deal with the analysis of commercial, mesh-based P2P streaming systems [6, 7, 8]. The goal of these studies is to understand the proprietary protocol used by the streaming system or to understand the effects of the networking environment and the user behavior. There are also several works in which solutions for mesh-based streaming are proposed and evaluated through simulations [9, 10, 11, 12, 13, 14]. These works often consider a different set of input parameters and performance measures, and therefore the proposed systems are hard to compare. There are however few works that address the streaming performance with analytic models. In [15] the minimum playback delay is derived for given source and peer transmission capacities, by considering the trade-off between the depth of the distribution trees and the time a peer needs to forward a packet to all of its downstream neighbors. The performance of BitTorrent-based streaming is addressed in [16], deriving relation between the achievable utilization of upload bandwidth and the number of segments or packets considered for sharing, which in turn is related to the playback delay peers will experience. Rate-optimal streaming

in complete meshes is considered in [17] using a queueing theoretic approach. The authors show that rate-optimal streaming is possible in a distributed manner and derive scheduling algorithms for the cases when the bottleneck is at the uplink or at the downlink of the peers. This work is extended in [18] where the authors propose different algorithms, and evaluate their delay-throughput performance in overlays forming a complete graph, using mean-field approximations. We extended this work by relaxing the assumption on complete overlay graph in [20]. We derived analytical expressions for the data propagation in a mesh-based overlay with random packet forwarding given that peers always have perfect knowledge of the buffer contents of their neighbors. The present work extends [20] by capturing the effect of outdated buffer map information at the sender peer, by considering a larger set of forwarding schemes and by including the case of overlays with node churn.

The scalability properties of P2P streaming systems are addressed in [19] by use of large deviations theory. The paper proves for a large set of node-to-node delay distributions that in order to keep the same playout probability, the playback delay has to be increased in proportion to the increase of the overlay path lengths. We use this result to assess the structure of the distribution trees, even if they are formed in a distributed manner through the local decisions of the transmitting peers.

3 System overview

We consider a peer-to-peer live streaming system consisting of a streaming server and N peers. The peers form an overlay such that each peer knows about up to d other peers, called neighbors, and about the streaming server. In the case of node churn, arriving peers become neighbors of d peers that have less than d neighbors. We assume that once a peer leaves the overlay its neighbors do not actively look for new connections but wait until a newly joining peer connects to them.

The system is time slotted, with one slot being equal to the packet content duration. At the beginning of each time slot a new packet is generated at the server and is forwarded to m randomly chosen peers. Peers distribute the data to their neighbors according to some scheduling algorithm. At the beginning of every time-slot every peer makes a forwarding decision, i.e., chooses a neighbor and a packet, and sends the chosen packet to the chosen neighbor. Packets that are sent at the beginning of a time-slot reach their destination at the end of the same slot. The upload capacity of peers is equal to the stream rate, i.e., one packet per slot, while their download capacity is unlimited, i.e., they can receive up to d packets from their neighbors in one slot.

Each peer maintains a playout buffer of B packets in which it stores the packets received from its neighbors before playing them out. A peer that joins the overlay before the beginning of time slot i starts playout with packet i at the beginning of time slot $i + B$. Consequently, the playback is synchronized among all peers, and both the startup delay and the playback delay are B time slots.

The nodes' decisions on forwarding a packet are based on the information they have about the packets that their neighbors possess in their playout buffers. Nodes know about the contents of their neighbors' playout buffers via exchanging buffer maps. In a scenario with *perfect information*, each peer would have full knowledge of the buffer contents of its neighbors upon taking a forwarding decision. However, such a scheme would be infeasible in a real peer-to-peer system due to the overhead related to the buffer map exchange among all the peers in the overlay. Consequently, we consider that nodes send updated buffer maps to their neighbors every u time slots, and hence, forwarding decisions are made based on *outdated information*. The shorter the update interval, the more accurate is the information in the buffer maps, but at the same time the higher is the overhead due to message exchange in the overlay.

3.1 Push-based forwarding schemes

In this paper we consider four push-based forwarding schemes. All schemes take as input the buffer maps of all the neighbors and return as output the forwarding decision, i.e., a (neighbor, packet) pair. Before describing the forwarding schemes, we define the notion of eligible packet and that of eligible neighbor. We say that a packet is eligible with respect to a neighbor if the peer has the packet in its playout buffer but the packet is not present in the most recently received buffer map from the neighboring peer. Similarly, we say that a neighbor of a peer is eligible if the peer has at least one eligible packet with respect to that neighbor.

Random peer - Random packet (RpRp): The peer constructs the set of eligible neighbors based on the buffer maps received from its neighbors, and picks uniformly at random one neighbor from the set. The peer then creates the set of eligible packets with respect to the chosen neighbor, and picks a packet uniformly at random from the set.

Determined peer - Random packet (DpRp): The peer calculates the number of eligible packets for all of its eligible neighbors based on the buffer maps. It selects a neighbor with a probability proportional to the number of eligible packets with respect to that neighbor. The packet to be sent to the selected neighbor is then chosen uniformly at random from the set of eligible packets with respect to that neighbor. By following this scheme a peer forwards data with higher probability to a neighbor to which it has many eligible packets.

Latest blind packet - Random useful peer (LbRu): The peer chooses the packet with the highest sequence number in its playout buffer independent of whether the packet is eligible with respect to any of its neighbors. It then chooses uniformly at random one of its neighbors that are missing the chosen packet. The unconditional selection of the packet implies that there might be cases when a peer cannot forward the last packet because all the neighbors already have it, while some neighbor might be missing a packet with lower sequence number that the forwarding peer has.

Latest useful packet - Useful peer (LuUp): The peer chooses the packet that has the highest sequence number among the packets that are eligible with respect to at least one neighbor. It then picks uniformly at random among the neighbors that are eligible with respect to the chosen packet. The chosen packet is not necessarily the last packet in the buffer, which is the difference of this scheme compared to the *LbRu* forwarding scheme.

We will refer to the two first schemes as random-packet schemes as they pick the packet to be forwarded at random, and we refer to the two latter schemes as fresh-data-first schemes because they always try to forward the data with the highest sequence numbers. *RpRp* was previously evaluated for the specific case of per slot buffer map exchange [20] and fresh-data-first schemes were analyzed in [18] for the case when the overlay forms a complete graph and with the assumption that local scheduling decisions at the transmitting peers are immediately known in the neighborhood of the receiving peer, that is, transmission of duplicates is fully avoided.

4 Analytical Model - Data dissemination

In this section we first describe the analytical model of an overlay without churn, then we show how this model can be used to approximate the effects of node churn. While the model considers a slot synchronized system, we argue that it describes well also the behavior of unsynchronized systems, referring to results in [20].

The key assumption of our model is that the contents of the playout buffers of neighboring peers are independent and statistically identical. The assumption is based on the observation that if peers maintain a fair number of neighbors then the local forwarding decisions will lead to per packet distribution trees that are very different, and as a result the position of the peers in the distribution trees is statistically the same. We validate this assumption by simulations in Section 5.

We number packets according to their transmission time at the server, i.e., packet j is transmitted in slot j. Packet j is played out at the beginning of slot $j + B$ at each peer. Since playout occurs at the beginning of the time slots, a peer can play out packet j if the packet is in the playout buffer of the peer by the end of time slot $j + B - 1$.

Let us denote by $\ell(i)$ the lowest packet sequence number that peers would potentially forward in time slot i. Since in slot i a peer would not forward the packet that is to be played out in that slot, $\ell(i) = max\{0, i - B + 1\}$. The highest packet sequence number that can be forwarded by any peer in slot i is $i - 1$, because only the source has packet i in slot i.

We denote by $\mathcal{B}_i^r \in 2^{\{\ell(i),\ldots,i-1\}}$ the set of packets that peer r has in its playout buffer at the beginning of slot i, i.e., the set of packets that it could potentially forward in slot i. In Figure 2 we show an example of a peer's playout buffer at the beginning of three consecutive time slots. The packet in the bold cell is the

packet that should be played out at the given slot. At the beginning of slot $i = 9$ the playout of the packets has not started and all the packets present in the buffer belong to the set $\mathcal{B}_i$. At the beginning of slot $i = 10$, packet 0 is to be played out, so it cannot be eligible for sending and the same holds for packet 1 at time $i = 11$.

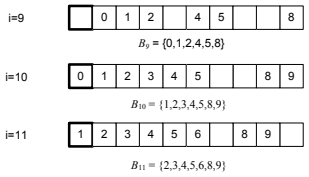


Figure 2: Example of a playout buffer of a peer and the set of packets that the peer could forward for three consecutive time-slots. Cell with bold edges contains the packet that is to be played out during the specific slot. In slot i=9 the peer receives packet 3 from a neighbor and packet 9 from the source. In slot i=10 it receives packet 6 from a neighbor.

Let us denote by P_i^j the probability that a peer is in possession of packet j by the end of time-slot i, i.e., $P(j \in \mathcal{B}_i) = P_{i-1}^j$. Hence, packet j will be successfully played out with probability P_{j+B-1}^j. Similarly, let us denote by $\overline{P}_i^j$ the probability that a peer possesses packet j at the end of slot i as known by one of its neighbors at the end of slot i. Since buffer map updates do not occur instantaneously, $\overline{P}_i^j = P_i^j$ does not necessarily hold. We consider that buffer map updates occur every u time slots, with the first exchange at the end of slot $i = 0$, so that we can express $\overline{P}_i^j$ as a function of P_i^j

$$\overline{P}_i^j = \begin{cases} 0 & u = \infty \\ P_{\lfloor i/u \rfloor u}^j & 1 \leq u < \infty. \end{cases} \tag{1}$$

In the following we derive recursive formulae to calculate the probability of successful playout in an overlay overlay without and with node churn.

4.1 Overlay without node churn

Let us consider an overlay in which peers do not join nor leave and there are no losses in the overlay links between peers. Our goal is to calculate the probability P_i^j for an arbitrary peer r in the overlay. The peer has packet j at the end of slot i either if it already had it at the end of slot $i-1$, or if it did not have it at the end of slot $i-1$ but received it from some neighbor s during slot i. This can be expressed by a general recursive equation introduced in [20]

$$P_i^j = \begin{cases} 0 & i < j \\ \frac{m}{N} & i = j \\ P_{i-1}^j + (1 - P_{i-1}^j)(1 - (1 - \pi_i^j)^d) & i > j. \end{cases} \tag{2}$$

where π_i^j is the probability that some neighbor s forwards packet j in slot i to peer r, given that peer r does not have packet j. P_j^j is the probability that a node receives the packet directly from the server. P_i^j is determined by π_i^j, i.e, it depends on the forwarding scheme used. In the following subsections we will derive expressions for the π_i^j for the *RpRp*, *DpRp*, *LbRu* and *LuUp* forwarding schemes. When describing the four schemes we refer to the peer that makes the forwarding decision as peer s, and to the peer that is to receive packet j as peer r.

4.1.1 Random peer - Random packet

In the *RpRp* scheme peer s first chooses a peer, and then it chooses a packet to be sent. Let us define the corresponding events

$$\begin{aligned} A &:= \{s \text{ has packet } j \text{ and } r \text{ does not have it}\},\\ C &:= \{s \text{ chooses to send a packet to } r\},\\ D &:= \{s \text{ chooses to send packet } j\}. \end{aligned}$$

Using these events the probability π_i^j can be written as

$$\begin{aligned} \pi_i^j &= P(s \text{ sends packet } j \text{ to } r | j \notin \mathcal{B}_i^r)\\ &= P(s \text{ sends packet } j \text{ to } r | j \notin \mathcal{B}_i^r, j \in \mathcal{B}_i^s) \cdot P(j \in \mathcal{B}_i^s)\\ &= P(C|A) \cdot P(D|A \cdot C) \cdot P(j \in \mathcal{B}_i^s). \end{aligned} \tag{3}$$

By definition $P(j \in \mathcal{B}_i^s) = P_{i-1}^j$, so we proceed to the calculation of $P(C|A)$. Under the *RpRp* scheme, peer s transmits to one of its eligible neighbors. Given the condition that node r is eligible, the probability that node s selects node r to transmit to is

$$P(C|A) = \sum_{k=0}^{d-1} \frac{1}{k+1} \binom{d-1}{k} p_e^k (1-p_e)^{d-k-1}, \tag{4}$$

where p_e is the probability that a neighbor is eligible

$$\begin{aligned} p_e = (1 - \overline{P}_{i-1}^j) + \overline{P}_{i-1}^j (1 - \\ \prod_{w=\ell(i), w\neq j}^{i-1} (1 - P_{i-1}^w (1 - \overline{P}_{i-1}^w))). \end{aligned} \tag{5}$$

Finally, we calculate $P(D|A \cdot C)$, the probability that s will send packet j and not another eligible packet to r. Let us denote by the r.v. K the number of packets that are eligible to be sent from s to r including packet j. Clearly, $K \geq 1$ because packet j is eligible with respect to node r. Then the probability that packet j is sent is

$$P(D|A \cdot C) = \sum_{k=1}^{i-\ell(i)} \frac{1}{k} P(K = k). \tag{6}$$

In the following define a recursion to calculate the distribution of K.

We define the probability vector

$$Q = \{P_i^{\ell(i)} \cdots P_i^{j-1}, 1, P_i^{j+1} \cdots P_i^{i-1}\},$$

which contains the probabilities that node s has packets at the different buffer positions, given that it has packet j. Similarly, we define the probability vector

$$\overline{Q}_r = \{\overline{P}_i^{\ell(i)} \cdots \overline{P}_i^{j-1}, 0, \overline{P}_i^{j+1} \cdots \overline{P}_i^{i-1}\},$$

which represents the information that the sending peer s has about the buffer contents of peer r, given that peer r does not have packet j.

We initialize the recursion by setting $L_{0,0} = 1$ and $L_{k,0} = 0$ for $k < 0$. The recursion is then defined by

$$\begin{aligned} L_{k,l} &= L_{k,l-1}(1 - Q(l)(1 - \overline{Q}_r(l))) + \\ &\quad L_{k-1,l-1}Q(l)(1 - \overline{Q}_r(l)). \end{aligned} \tag{7}$$

The recursion is executed for $l = 1 \ldots i - \ell(i)$ and for every l for $k = 0 \ldots l$. The distribution of K is then given as $P(K = k) = L_{k,i-\ell(i)}$ for for $k = 1 \ldots i - \ell(i)$. By plugging (4) and (6) into (3) we obtain the probability π_i^j.

4.1.2 Determined peer - Random packet

Similar to the *RpRp* forwarding scheme, and using the events introduced there, we can write

$$\begin{aligned} \pi_i^j &= P(s \text{ sends packet } j \text{ to } r | j \notin \mathcal{B}_i^r) \\ &= P(s \text{ sends packet } j \text{ to } r | A) \cdot P(j \in \mathcal{B}_i^s). \end{aligned} \tag{8}$$

Nevertheless, for the *DpRp* forwarding scheme the choice of a peer and the choice of a packet are correlated. Hence we calculate the joint probabilities directly.

Let us denote by the r.v. K the total number of forwarding decisions that node s can make with respect to any of its neighbors (i.e., node-packet pairs), given that s possesses j and r does not possess j. Clearly, $K \geq 1$ because packet j is eligible with respect to node r. Given the distribution of K the conditional probability that packet j is sent to r is expressed as

$$P(s \text{ sends packet } j \text{ to } r | A) = \sum_{k=1}^{d(i-\ell(i))} \frac{1}{k} P(K = k). \tag{9}$$

In the following we define a recursion to calculate the distribution of K. Apart from the vectors Q and $\overline{Q}_r$ defined in the previous section, we define the probability vector

$$\overline{Q} = \{\overline{P}_i^{\ell(i)} \cdots \overline{P}_i^{i-1}\},$$

which represents the information that the sending peer s has about the buffer contents of its neighboring peers apart from node r.

As before, we initialize the recursion by setting $L_{0,0} = 1$ and $L_{k,0} = 0$ for $k \neq 0$. The recursion is however defined by

$$\begin{aligned} L_{k,l} = & L_{k,l-1}\left\{(1-Q(l)) + Q(l)\overline{Q}(l)^{d-1}\overline{Q}_r(l)\right\} \\ + & \sum_{z=1}^{d-1} L_{k-z,l-1}Q(l)\left\{\left(\begin{array}{c} d-1 \\ z \end{array}\right)\overline{Q}(l)^{d-1-z}(1-\overline{Q}(l))^z\overline{Q}_r(l) \right. \\ & \left. + \left(\begin{array}{c} d-1 \\ z-1 \end{array}\right)\overline{Q}(l)^{d-z}(1-\overline{Q}(l))^{z-1}(1-\overline{Q}_r(l))\right\} \\ + & L_{k-d,l-1}\left\{Q(l)(1-\overline{Q}(l))^d(1-\overline{Q}_r(l))\right\}. \end{aligned} \tag{10}$$

The recursion is executed for $l = 1, \ldots i - \ell(i)$, and for every l for $k = 0 \ldots ld$. The distribution of K is then given by $P(K = k) = L_{k,i-\ell(i)}$ for $k = 1 \ldots (i - \ell(i))d$.

4.1.3 Latest blind packet - Random useful peer

The probability that r will receive packet j from s in time slot i is equal to the probability that j is the latest packet in the buffer of s, times the probability that s will choose r among its neighbors that do not have packet j. We define the events

$$\begin{aligned} E &:= \{j \text{ is the latest packet in the buffer of } s\}, \text{ and} \\ F &:= \{s \text{ chooses to send packet } j \text{ to } r\ \}. \end{aligned}$$

The expression for π_i^j can then be written as

$$\pi_i^j = P(E) \cdot P(\mathrm{F}|\mathrm{E}\cdot\mathrm{A}) \cdot P(j \in \mathcal{B}_i^s). \tag{11}$$

Again we have $P(j \in \mathcal{B}_i^s) = P_{i-1}^j$, and proceed to the calculation of $P(E)$. Event E occurs if there are no packets in the buffer of s that have a higher sequence number than j

$$P(E) = \prod_{v=j+1}^{i-1} P_{i-1}^v. \tag{12}$$

Finally, we calculate the probability $P(\mathrm{F}|\mathrm{E}\cdot\mathrm{A})$. We define the r.v. K as the number of neighbors of s other than r that are missing packet j. Then, we get that

$$\begin{aligned} P(F|E\cdot A) &= \sum_{k=0}^{d-1} \frac{1}{k+1} \cdot P(K=k) \\ &= \sum_{k=0}^{d-1} \frac{1}{k+1} \cdot \binom{d-1}{k} (1-(\overline{P}_i^j))^k \cdot (\overline{P}_i^j)^{d-1-k}. \end{aligned} \tag{13}$$

Combining the above equations we get the probability π_i^j.

4.1.4 Latest useful packet - Random useful peer

In addition to the events defined in the previous subsections let us define the event

$$G := \{j \text{ is the latest useful packet in the buffer of } s\}.$$

Using the same rationale as in the previous case, we can express π_i^j as

$$\begin{aligned} \pi_i^j &= P(s \text{ sends pkt } j \text{ to } r | j \notin \mathcal{B}_i^r) \\ &= P(s \text{ sends pkt } j \text{ to } r | A) \cdot P(j \in \mathcal{B}_i^s) \\ &= P(C|A \cdot G) \cdot P(G) \cdot P(j \in \mathcal{B}_i^s). \end{aligned} \tag{14}$$

Again, we have $P(j \in \mathcal{B}_i^s) = P_{i-1}^j$, so we turn to the probabilities $P(G)$ and $P(D|A \cdot G)$. Packet j is latest useful if there is no packet with higher sequence number that some of the neighbors of s are in need of

$$P(G) = \prod_{v=j+1}^{i-1} (1 - P_{i-1}^v (1 - (\overline{P}_{i-1}^v)^d)). \tag{15}$$

Then we calculate the probability $P(C|A \cdot G)$, that s will choose r to send the packet to, among all its neighbors

$$\begin{aligned} P(C|G \cdot A) = \sum_{k=0}^{d-1} \frac{1}{k+1} \cdot \\ \binom{d-1}{k} (1 - (\overline{P}_i^j))^k \cdot (\overline{P}_i^j)^{d-1-k}. \end{aligned} \tag{16}$$

Plugging equations (15) and (16) into (14) yields the probability π_i^j. Note that the complexity of the presented models does not depend on the overlay size N, and therefore they provide excellent tools to evaluate the scalability of the scheduling schemes.

4.2 Overlays in the presence of node churn

Node-churn can affect the performance of a push-based system in three ways. First, if a peer does not know that some of its neighbors departed from the overlay, it might forward packets to non-existing neighbors, which leads to a loss of forwarding capacity. We do not explicitly consider this artifact, as it can be easily modelled as the decrease of the forwarding rate of the peers. Second, a peer that arrives before the beginning of time slot i starts playback with packet i at time slot $i + B$, consequently the peer does not request packets with a sequence number lower than i. Third, the number of neighbors a peer has is not constant, but varies over time as nodes join and leave. In the following we show how to estimate the effects of the change of the number of neighbors on the overlay's performance.

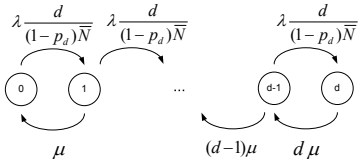


Figure 3: Markov process used to model the evolution of the number of neighbors of a peer.

For simplicity we consider that nodes arrive to the overlay according to a Poisson process with intensity λ and their lifetime follows an exponential distribution with mean $1/\mu$. Each peer joining the overlay is assigned d neighbors uniformly at random from the nodes that have less than d neighbors. Let us denote by the r.v. $D(t)$ the number of neighbors of a peer at time t ($D(t) \in \{0, \ldots, d\}$, $t \in [0, \infty)$) and by the r.v. D_i the number of neighbors of a peer at the beginning of time slot i ($D_i \in \{0, \ldots, d\}$). In the following we derive the distribution of D_i in the steady state of the system, which is the same as the steady state distribution of $D(t)$, i.e., the probabilities $p_z = \lim_{t\to\infty} P(D(t) = z)$, $z \in \{0, \ldots, d\}$.

The average number of peers in the overlay in steady state is $\overline{N} = \lambda/\mu$ and we can approximate the evolution of $D(t)$ with a one dimensional continuous time Markov process as shown in Figure 3. We can approximate the arrival rate of the neighbors to a peer already in the overlay by $\lambda \cdot d/(\overline{N}(1 - p_d))$, i.e., in the denominator we use the average of the number of peers with less than d neighbors instead of its actual value. We use an iterative method to calculate the steady state distribution of the Markov process: we start with an initial value of $p_d = 0$ to derive the next value of the steady state probability of p_d until the value converges.

The evolution of P_i^j for an arbitrary peer r depends on the number of neighbors of the node itself and on π_i^j, which in turn depends on the number of neighbors of the neighboring peers of node r. The exact calculation of P_i^j has a complexity of d^d and is computationally not feasible. Hence we make three approximations. First, we approximate the number of neighbors of all the neighboring peers of r with $E[D_i]$ when calculating π_i^j. Second, we assume that the content of the playout buffer of the arriving nodes is statistically identical to that of the nodes already present in the overlay, i.e., can be described by P_i^j. Third, to calculate the probability that the peer receives a packet directly from the root we use $\overline{N}$ instead of the actual overlay size in time-slot i. We evaluate the effect of these approximations in Section 5 by simulation.

The distribution of the number of neighbors of peer r in an arbitrary slot is given by the steady state distribution of D_i, hence for an arbitrary peer in a dynamic

overlay instead of (2) we can write

$$P_i^j = \begin{cases} 0 & i < j \\ m/\overline{N} & i = j \\ \sum_{z=0}^{d} P(D_i = z) \cdot \{P_{i-1}^j + & \\ \quad (1 - P_{i-1}^j) \cdot (1 - \pi_i^j)^z\} & i > j \end{cases} \tag{17}$$

5 Performance Evaluation

In the following we first validate our modelling assumptions via simulations, then we evaluate the performance of the four forwarding schemes based on analytical and simulation results.

5.1 Simulation methodology

For the simulations we developed a packet level event-driven simulator. We construct the overlay as follows. Each peer wishing to join sends a connection request message to the boot-strap node, which may be the source node. The boot-strap node responds with a list of randomly selected peers that are already part of the overlay. The joining peer contacts the peers in this list in order to establish neighbor relationships with at least d_{min} but not more than d of them, where $d_{min} = 0.85d$. If it cannot find d_{min} neighbors, the peer issues a new request to the boot-strap node. If, after the second request, the peer still cannot connect to d_{min} neighbors, it issues a force connection request to random peers in the overlay; those peers are forced to drop one of their neighbors and replace it with the joining peer.

For the simulations we first construct an overlay with N peers, thereafter the data distribution starts. For simulations with node churn, nodes join the overlay according to a Poisson process, and the life time distribution is exponential. We run the simulations for 2000 time slots, and the simulation results shown are the averages of 10 simulation runs.

5.2 Model validation

As a first step we validate our assumption that buffer maps are statistically identical, which means that all peers have the same probability of possessing any particular packet at any time. Instead of evaluating the entire buffer map, we compare the probabilities that a packet is played out successfully, that is, it is received by the playout time.

In Figure 4 we show the histogram of the probability of successfully played out packets across the peers, considering the four forwarding schemes and the case of perfect information ($u = 1$). The vertical line shows the mean value of the play-out probability averaged over all peers, while in the legend we show its standard

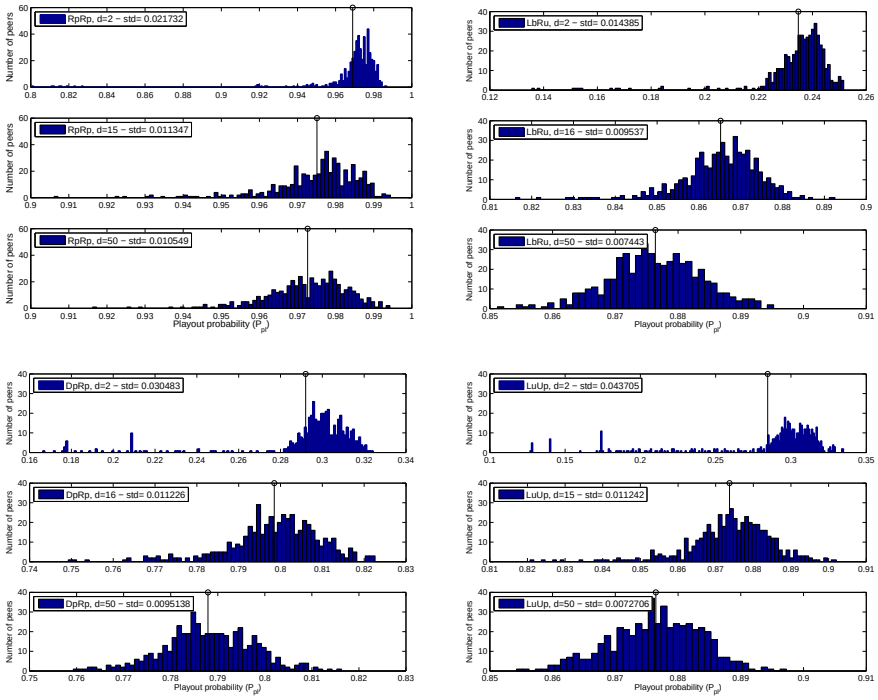


Figure 4: Histogram of the playout probability for the considered forwarding schemes for an overlay of $N = 500$, $m = 11$, $B = 40$ and for $d = 2, 15$ and 50 respectively.

deviation. When the number of neighbors of a peer is very low, such as $d = 2$, the probabilities are dispersed over a wide range, as can be seen both visually as well as by the standard deviation. It means that our assumption on statistically identical buffer maps does not hold. As d increases, the probabilities for different peers approach the mean. This shows that for reasonably high values of d our modelling assumption holds and the results obtained by the model should be accurate. Figure 5 shows the histogram of the playout probability for the four algorithms and for the case where nodes have outdated information about their neighbors' playout buffers with parameters $u = 4$ and $d = 50$. Compared to the perfect information case the standard deviation is higher for all schemes. We see significant effect of delayed buffer maps in the case of fresh-data-first schemes, where the initial statistical difference among the playout buffers is amplified due to the rather deterministic forwarding decisions.

To investigate the impact of the neighborhood size, d, on the overlay performance, we show in Figure 6 the probability of successful play out versus the number of neighbors d for all considered schemes. The figure contains both analytical and simulation results. For small values of d there is a discrepancy between the model

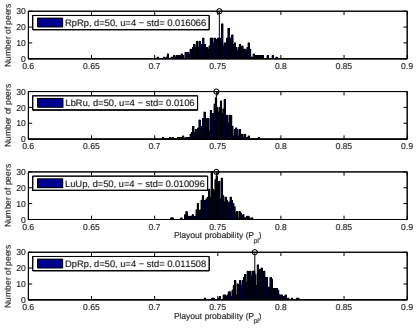


Figure 5: Histogram of the playout probability in the presence of delayed buffer map updates. Overlay of $N = 500$, $m = 11$, $B = 40$ and $u = 4$.

and the simulations, since in this region the assumption on statistically identical buffer maps does not hold. Nevertheless, as d increases the analytical results approach the simulation results. For the case of perfect information, $u = 1$, and for values of $d > 10$ the approximation is very good for all schemes, whereas for $u > 1$ the analytical and the simulations results converge at a higher value of d. We experience slower convergence in the case of fresh-data-first schemes as expected from the standard deviation values on Figure 5.

Based on the figures we can also draw the important conclusion that the playout probability is insensitive to the increase of the neighborhood size above a certain value of d. This suggests that a peer can reach the best achievable performance in terms of playout probability by maintaining a reasonable number of neighbors, and a slight variation of d caused by churn would not lead to variations in the performance.

5.3 Playout probability and playback delay

Next, we investigate the performance of the four forwarding schemes in terms of the ratio of successfully played out packets for static overlays. We consider an overlay of size $N = 500$, and the number of neighbors is $d = 50$. In Figure 7 we show the ratio of successfully played out packets versus the playback delay in the case of perfect information. In all the cases, we can see that the analytical results are very close to the simulation results. For the fresh-data-first and the *DpRp* schemes there is a perfect match, while for *RpRp* the model slightly underestimates the playout probability for small playback delays.

For small buffer lengths the fresh-data-first schemes perform considerably better than the random-packet schemes, which is in accordance with the results presented in [18, 21] and confirms the low diffusion delays achieved by the fresh-data-first

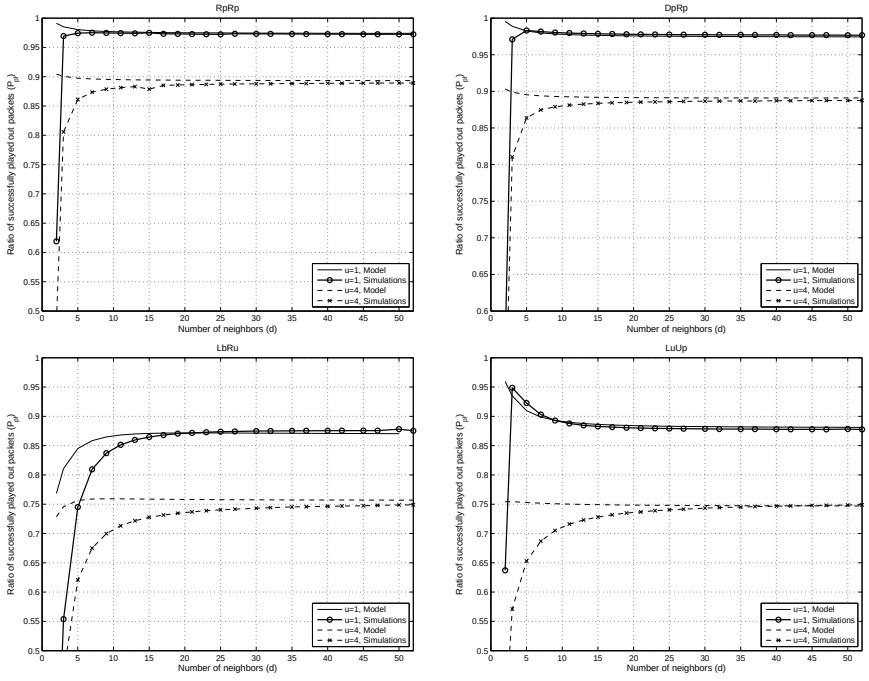


Figure 6: Probability of successfully played out packets vs number of neighbors. Overlay with $N = 500$, $m = 11$, $B = 40$. Model and simulations.

schemes. The ratio of successfully played out packets remains however low for *LbRu* and *LuUp* even at high playback delays. The reason of this limited performance is the packet selection scheme: since only fresh packets are considered for transmission, the probability that more than one neighbor of a peer transmits the same packet in the same time slot is relatively high and is not affected by the size of the playout buffer. This result does not contradict the results of [18], since there the same schemes are considered assuming that scheduling decisions are shared without delay. As collision is avoided with this assumption, the playout probability reaches one.

The playout probability increases with the playback delay for the *RpRp* and the *DpRp* schemes, and converges to one. In these cases the randomness of packet selection increases with B and consequently the probability of peers sending the same packet to the same peer in the same time slot approaches zero.

The *LbRu* and *LuUp* schemes yield almost the same performance for all buffer length values, except for large values of B, when the performance of *LuUp* is marginally better. Similarly, *DpRp* slightly outperforms *RpRp* for large values of B as it makes maximum use of the increase of the set of eligible packets.

Next we investigate why data diffusion is slower in the random-packet schemes.

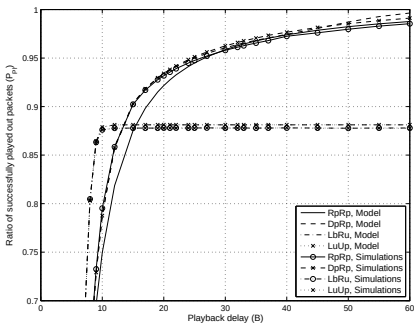


Figure 7: Ratio of successfully played out packets vs playback delay for four forwarding schemes. Overlay with $N = 500$, $m = 11$, $d = 50$ and $u = 1$. Model and simulations.

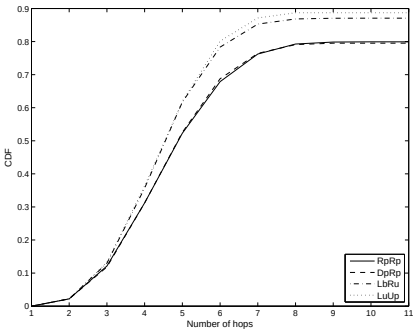


Figure 8: CDF of the number of hops from the source. Overlay with $N = 500$, $m = 11$, $d = 50$, $B = 10$ and $u = 1$. Simulations.

Figure 8 shows the cumulative distribution function of the number of overlay hops from the source node for a specific packet. The results shown here are obtained by tracking one randomly selected packet in 10 simulation runs over 10 different overlays. We observe that using the fresh-data-first schemes peers receive packets after somewhat fewer hops, which indicates that the trees that packets traverse to reach the peers have slightly higher fanout. This small difference alone does not explain the better delay performance of fresh-data-first schemes. The main reason is that in these schemes the time between receiving a packet and forwarding it to a neighbor is shorter as well. Nevertheless, the random-packet schemes might disseminate the data at a lower pace, but at relaxed playout delay constraints it reaches a larger portion of the peers in the overlay.

5.4 Playout probability and outdated buffer maps

Intuitively, the outdated information contained in a buffer map leads to sub-optimal forwarding decisions, and hence to the decreased efficiency of the forwarding algorithms. We show results that confirm this intuition in Figure 9 for *RpRp* and *LbRu*. We observe that the curves that correspond to the case of outdated information ($u > 1$) are similar in shape to the case with perfect information ($u = 1$), but the playout probability is always lower. For small playback delays the impact of outdated information seems to be rather small, however at larger delays the difference becomes significant. Figure 10 shows how fast the playout probability converges to the lower bound, when scheduling decisions are made without buffer map information (that is when $u \to \infty$ in the analytical model). The playout prob-

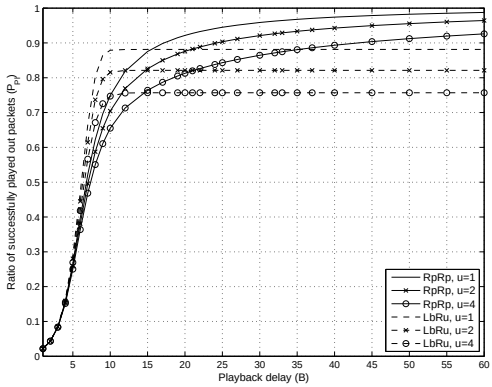


Figure 9: Ratio of successfully played out packets vs playback delay in the presence of outdated buffer maps. Overlay with $N = 500$, $m = 11$, $d = 50$. Model.

ability decreases fast with the increase of u, which indicates that the efficiency of the forwarding algorithms is very sensitive to the timeliness of the buffer content information. The faster decrease in the case of *LbRu* is again due to the fresh-data-first nature of the scheme. Between buffer map updates information about older packets is still available in the buffer, but those old packets are not considered for transmission.

5.5 Scalability

In this subsection we evaluate the scalability of the considered schemes in terms of the overlay's size. Figure 11 shows the minimum playback delay required to achieve a playout probability of 0.85 as a function of the overlay size N for the *RpRp* and the *LbRu* schemes. The *DpRp* and the *LuUp* schemes give similar results. The horizontal axis is in logarithmic scale. We see that the increase of the necessary playback delay is proportional to $log(N)$ for both schemes and even for $u > 1$ if the considered playout probability is achievable. In [19] the authors showed that the increase of the playback delay required to maintain the playout probability unchanged is proportional to the increase of the depth of the overlay. Hence, we can conclude that all the considered push-based scheduling schemes lead to data distribution trees with a depth that is logarithmic in the number of peers in the overlay.

5.6 Effects of node churn

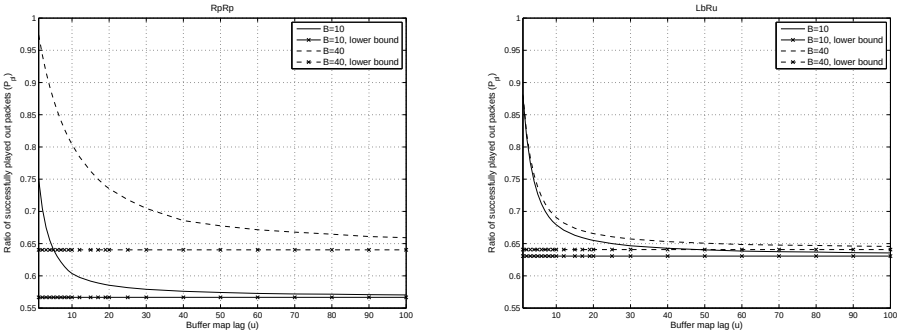


Figure 10: Ratio of successfully played out packets vs buffer map update lag. Overlay with $N = 500$, $m = 11$, $d = 50$. *RpRp* and *LbRu* schemes. Model.

In this subsection we evaluate the effects of node churn on the system performance. Figure 12 shows the playout probability as a function of the playback delay for the *RpRp* and the *LbRu* schemes with and without node churn. The figure shows that the playout probability is only slightly affected by node churn. This result is in accordance with Figure 6: node churn does not decrease the performance of the overlay as long as peers are able to maintain a certain number of neighbors. A consequence of the insensitivity to the evolution of the number of neighbors is that the assumption on the node life time distribution does not affect our results (the steady state probability distribution of the M/G/∞ queue is known to be insensitive to the service time distribution).

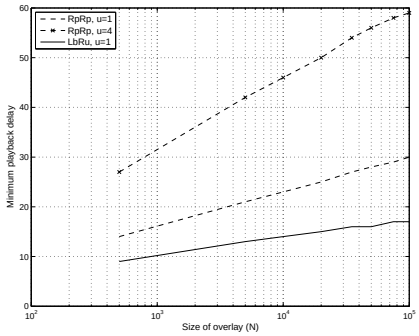


Figure 11: Minimum playback delay required to achieve a playout probability of 0.85 versus overlay size. *RpRp* and *LbRu* schemes. Model.

Surprisingly however, the simulation results show that the overlay may perform slightly better in the presence of node churn. Clearly, the improved performance cannot be a consequence of the variation of the number of neighbors of the nodes. In order to understand the reason for the improved performance we ran simulations with *altruistic* peers. An *altruistic* peer behaves slightly different than the *conservative* peers considered until now. An altruistic peer that arrives before the beginning of slot i requests packets with a sequence number at least $i - B/2$ from its neighbors in order to help the data distribution, even though itself starts playout only at slot $i + B$ with packet i.

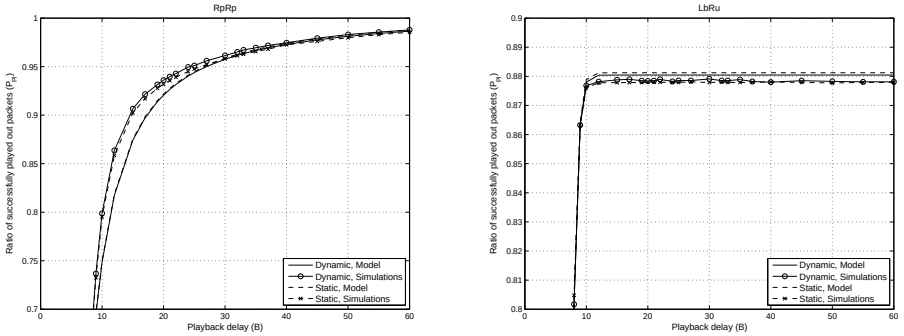


Figure 12: Ratio of successfully played out packets vs playback delay under churn. Overlay with N=500, m=11 and d=50, $\frac{1}{\mu}$=50 slot. Model and Simulations.

The results of the simulations performed with the *altruistic* peers and the *conservative* peers can be seen in Figure 13, which shows the playout probability as a function of the mean lifetime of peers measured in time-slots. The static case is shown as reference. The lower the mean lifetime, the higher the churn rate, i.e., the more dynamic is the scenario. Contrary to what one would expect, the overlay may benefit from high churn rates if peers are *conservative*, but high churn leads to decreased performance if peers are *altruistic*. We explain this phenomenon by considering an arbitrary neighbor s of a *conservative* peer that joins the overlay at slot i. The peer is interested in packets that are generated at or after slot i. Consequently, the neighboring nodes of s that were already in the overlay have fewer contestants for the packets that were generated before slot i, and may get those packets sooner.

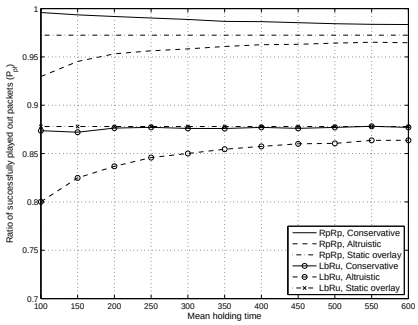


Figure 13: Ratio of successfully played out packets versus mean holding time for the *RpRp* and *LbRu* schemes. Simulations.

6 Conclusion

In this paper we proposed an analytic framework to assess the performance of various scheduling algorithms for mesh-based P2P streaming systems. The modelling framework is based on the observation that with peers maintaining a fair number of neighbors, the local scheduling decisions may generate per packet distribution trees that are very different and as a result the position of the peers in the trees is statistically the same. Using this framework, we derived analytic performance measures for four forwarding schemes. We proved that fresh-data-first schemes, though able to diffuse data fast, have a limit on the ratio of peers that they can deliver data to. In contrast, the random-packet schemes can achieve a good playout probability and playback delay trade-off. We also evaluated the performance of the forwarding algorithms under outdated information and showed that outdated information quickly leads to a significant decrease of the system's performance. Furthermore, we developed a model that shows that in a dynamic overlay the variation of the number of the peers' neighbors does not affect the overlay performance as long as peers are able to maintain a fair number of neighbors. We also showed that in a dynamic environment, newly arriving nodes can actually act in a beneficial way for the system, by properly adjusting their initial interest window: node arrivals increase slightly the average playout probability compared to a static overlay with the same characteristics. The analytic framework of this paper can be used to evaluate various forwarding solutions and their combinations, and also the effect of network parameters. In our future work we will extend the framework to propose and evaluate scheduling solutions for overlays with heterogeneous uplink and downlink capacities.

References

[1] PPLive, http://www.pplive.com/en/about.html.

[2] Octoshape, http://www.octoshape.com.

[3] UUSee, http://www.uusee.com.

[4] X. Hei, Y. Liu, and K. Ross. IPTV over p2p streaming networks: The mesh-pull approach. *IEEE Communications Magazine*, 46(2):86–92, February 2008.

[5] V. Fodor and Gy. Dán. Resilience in live peer-to-peer streaming. *IEEE Communications Magazine*, 45(6):116–123, June 2007.

[6] S. Birrer and F. Bustamante. A comparison of resilient overlay multicast approaches. *IEEE Journal on Selected Areas in Communications*, 25:1695–1705, December 2007.

[7] B. Li, S. Xie, G.Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang. An empirical study of the coolstreaming+ system. *IEEE Journal on Selected Areas in Communications*, 25:1627–1639, December 2007.

[8] X. Hei, C. Liang, Y. Liu, and K.W. Ross. A measurement study of a large-scale p2p IPTV system. *IEEE Transactions on Multimedia*, 9:1672–1687, December 2007.

[9] R. Rejaie N. Magharei. Prime: Peer-to-peer receiver-driven mesh-based streaming. In *Proc. of IEEE INFOCOM*, 2007.

[10] X. Zhang, J. Liu, B. Li, and T-S. P. Yum. Coolstreaming/donet: a data driven overlay network for efficient live media streaming. In *Proc. of IEEE INFOCOM*, 2005.

[11] K. Nahrstedt J. Liang. Dagstream: Locality aware and failure resilient peer-to-peer streaming. In *Multimedia Computing and Networking (MMCN)*, 2006.

[12] Y. Tang M. Zhang, L. Zhao, J-G. Luo, and S-Q. Yang. Large-scale live media streaming over peer-to-peer networks through the global internet. In *Proc. ACM Workshop on Advances in peer-to-peer multimedia streaming (P2PMMS)*, 2005.

[13] L. Bracciale, F. Lo Piccolo, D. Luzzi, S. Salsano, G. Bianchi, and N. Blefari-Melazzi. A push-based scheduling algorithm for large scale p2p live streaming. In *Proc. of QoS-IP*, 2008.

[14] M. Faloutsos A. Vlavianos, M. Iliofotou. Bitos: Enhancing BitTorrent for supporting streaming applications. In *Proc. of IEEE INFOCOM*, 2006.

[15] Y. Liu. On the minimum delay peer-to-peer streaming: how realtime can it be? In *Proc. of MM'07*, 2007.

[16] S. Tewari and L. Kleinrock. Analytical model for bittorrent-based live video streaming. In *Proc. of IEEE NIME 2007 Workshop*, 2007.

[17] L. Massoulie et al. Randomized decentralized broadcasting algorithms. In *Proc. of IEEE INFOCOM*, 2007.

[18] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: Optimal performance trade-offs. In *Proc. of ACM SIGMETRICS*, 2008.

[19] Gy. Dán and V. Fodor Delay Asymptotics and Scalability for Peer-to-peer Live Streaming In IEEE Trans. on Parallel and Distributed Systems,to appear

[20] V. Fodor and I. Chatzidrossos. Playback delay in mesh-based peer-to-peer systems with random packet forwarding. In *Proc. of 1st International Workshop on Future Multimedia Networking (FMN)*, 2008.

[21] C. Liang, Y. Guo, and Y. Liu. Is random scheduling sufficient in p2p video streaming? In *Proc. of the 28th International Conference on Distributed Computing Systems (ICDCS)*, 2008.

Paper C

On the Effect of Free-riders in P2P Streaming Systems

Ilias Chatzidrossos and Viktória Fodor.
In Proceedings of the 4th International Workshop on QoS in Multiservice IP Networks (QoS-IP), Venice, Italy, February 2008.

On the Effect of Free-riders in P2P Streaming Systems

Ilias Chatzidrossos and Viktória Fodor

Wireless@KTH, KTH, Royal Institute of Technology
Osquldas vag 10, 100 44 Stockholm, Sweden
iliasc@ee.kth.se
vfodor@ee.kth.se

Abstract

Peer-to-peer applications exploit the users willingness to contribute with their upload transmission capacity, achieving this way a scalable system where the available transmission capacity increases with the number of participating users. Since not all the users can offer upload capacity with high bitrate and reliability, it is of interest to see how these non-contributing users can be supported by a peer-to-peer application. In this paper we investigate how free-riders, that is, non-contributing users can be served in a peer-to-peer streaming system. We consider different policies of dealing with free-riders and discuss how performance parameters such as blocking and disconnection of free-riders are influenced by these policies, the overlay structure and system parameters as overlay size and source upload capacity. The results show that while the multiple-tree structure may affect the performance free-riders receive, the utilization of the transmission resources is still comparable to that of an optimized overlay.

1 Introduction

A media streaming system following the peer-to-peer approach is based on the collaboration of users, that is, their willingness to contribute with their own resources to achieve common welfare. In a peer-to-peer system users form an application layer overlay network. Users joining the overlay can receive media streams "from the overlay" and distribute data they possess "to the overlay". Specifically, it means that users build up point-to-point supplier-receiver relations to distribute the media stream. As a result, the availability of the stream and the transmission resources increase with the number of users in the overlay. This makes the system scalable.

While many of the users in a peer-to-peer streaming system are willing to collaborate and offer their upload capacity to serve other peers, there might be users

that do not contribute to the transmission resources of the overlay. These users are commonly called as free-riders. While there are various solutions to make free-riders collaborate [1, 2], it might be of interest of peer-to-peer systems to serve the ones who have not enough upload capacity to contribute to the system resources significantly, or are not reliable enough to serve as supplier, like residential users with asymmetric connection [3] or mobile users connected through wireless links.

In this paper we discuss different policies an overlay can follow when serving these goodwill free-riders. We evaluate the system performance under churn, that is when nodes join and leave the overlay dynamically during the streaming session. First we consider optimized overlays when all transmission capacity is available for the users and evaluate the system performance based on Markovian models. Then we evaluate how the performance characteristics of the overlay changes if tree based streaming is applied.

A similar analysis is presented in [4] considering an overlay with high and low contributors. It is assumed that all users are accepted by the overlay and share the available capacity equally, that is, the overlay is optimized after each peer arrival or departure. It is shown that under churn the overlay performance in terms of delivering the full stream to all the users can be characterized by a critical ratio of high contributors. If the ratio of high contributors is above the critical value the system performs well, otherwise the system performs poor. This critical value is sensitive to the overlay size and to the playback delay.

Our work contributes to these results by considering a set of policies to deal with free-riders and comparing the performance of optimized and more realistic overlays. The rest of the paper is organized as follows. In Section 2 we overview some of the studies closely related to this work. Section 3 describes the policies we consider to control the number of free-riders in the streaming system. In section 4 we describe the Markovian model of optimized overlays. Section 5 describes the multiple-tree overlay we consider for comparison. Section 6 evaluates how the streaming performance changes in the case of push based streaming with rigid tree structures. Finally, Section 7 concludes our work.

2 Related Work

The architectures proposed for peer-to-peer streaming generally fall into one of two categories: push based or pull based [5]. First, the push method was proposed for live media streaming in [6, 7]. It follows the traditional approach of IP multicast, as streaming data is forwarded along multiple disjoint transmission trees, with the source of the stream as the root.

The pull method (also called swarming) was proposed recently, and aims at efficient streaming in highly heterogeneous or dynamic environments [8, 9, 10]. It follows the approach of off-line peer-to-peer content distribution. There is no global structure maintained, and neighbor relations are determined locally on the fly as

the overlay membership or the network environment changes. Peer nodes consider the reception of a block of consecutive packets (or larger segments) at a time and schedule to pull those packets from some of their neighbors.

The first analytical study of peer-to-peer streaming presented in [11] considers the delay performance of single tree architectures. The loss and delay characteristics of multiple-tree based systems is evaluated in [12, 13]. The first mathematical models of pull based systems are given in [14, 15]. An asymptotic model of optimized overlays, that can be applied for both pull and push based systems, also addressing the effect of low contributors is presented in [4].

3 Policies for Admitting Free-riders

In this paper we consider resource sharing policies where nodes receive always with full streaming rate. Instead, peers that do not offer significant upload capacity are admitted to the system only if there is available transmission capacity in the overlay, otherwise are denied to join the streaming session. For simplicity we will consider high contributors that offer upload capacity that is multiples of the streaming rate and free-riders that do not offer any upload capacity. Specifically, we evaluate two policies to control the participation of free-riders in the streaming sessions.

With the **Block and Drop (BD)** policy free-riders that would like to join the streaming session are blocked if the free upload capacity in the overlay is less than the streaming rate. Once admitted, the same users can be dropped if the overlay capacity decreases due to the consecutive departure of contributors. All peers in the overlay receive with full streaming rate. The BD policy can be characterized by the probability that a free-rider will be blocked when it attempts to join the system, and by the probability that once admitted the free-rider will be dropped from the overlay.

Under the **Block and Wait (BW)** policy free-riders are blocked if the overlay does not have enough available capacity as in the case of BD. The same users can be temporarily disconnected and have to wait to reconnect if there is not enough capacity to serve all peers. All connected peers receive with full streaming rate. We characterize the BW policy by the blocking probability of free-riders and the average number of free-riders waiting to reconnect.

Note that blocking, dropping and disconnecting free-riders do not require sophisticated admission control in an overlay. A contributor can always connect to the overlay by taking the position of a peer (or peers depending on the overlay topology) already in the overlay and serve those peers with its own upload capacity. Instead, free-riders can join or reconnect to the overlay only if there are contributors with not fully utilized upload capacity. Otherwise free-riders are blocked or dropped, alternatively temporarily disconnected.

4 The Performance of Optimized Overlays

We call an overlay optimized if it can utilize all upload capacity. This in turn may require that the overlay is reorganized at each node arrival and departure. In this section we evaluate the performance free-riders receive in such optimized overlays under churn.

We consider the stationary state of the system, when the arrival and departure rates are equal. We assume that the interarrival times of peers are exponentially distributed, this assumption is supported by several measurement studies [16, 17]. We assume that contributors and free-riders arrive independently. We approximate the session holding times of contributors and free-riders by the same exponential distribution. The distribution of the session holding times was shown to fit the log-normal distribution [16], however, using the exponential distribution makes modeling easier. We evaluate the effect of this assumption via simulations.

We denote the arrival intensity by λ and consider the system performance as α the ratio of free-rider users changes. The arrival intensity of contributors and free-riders is $\lambda_c = (1-\alpha)\lambda$ and $\lambda_f = \alpha\lambda$ respectively. The mean session holding times are $1/\mu = 1/\mu_c = 1/\mu_f$. The offered load in the overlay is $A = \lambda_c/\mu_c$, $A_c = (1-\alpha)A$ and $A_f = \alpha A$.

For simplicity, we assume that all contributors offer the same upload capacity that supports the transmission of c copies of the stream (we say the contributors upload capacity is c). The source of the stream can have different (often significantly higher) upload capacity and can transmit m copies of the stream. We call parameter m source multiplicity.

4.1 Overlay feasibility

As it is shown in [4], it is possible to construct an overlay for a static set of peers if the sum of the reception rates at the peers does not exceed the sum of the upload capacities of the source and all the peers. This feasibility constraint holds if the stream can be divided to arbitrary number of sub-streams and the nodes can maintain a very high number of neighbors.

For the BD and BW policies the feasibility constraint in [4] limits N_f the number of free-riders, given the number of contributors N_c in the overlay as

$$N_f \leq m + (c-1)N_c. \tag{1}$$

We define the feasibility limit for the overlay with churn similarly as

$$A_f \leq m + (c-1)A_c. \tag{2}$$

4.2 Overlay evolution

We model the evolution of the optimized overlay under the BD and BW policies with a Markov-chain. The system state has to reflect not only the number of nodes in the

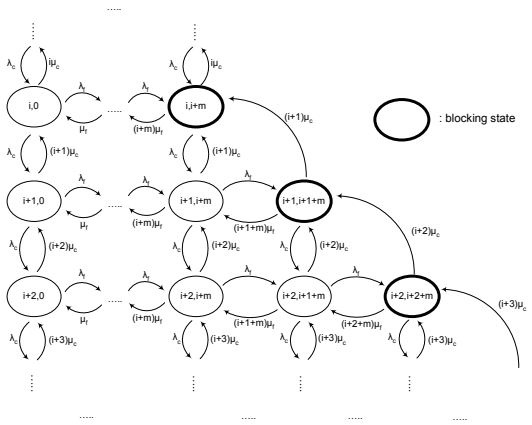


Figure 1: Markov-chain describing the overlay under the Block and Drop policy

overlay, but also the capacity available for arriving nodes. We have chosen to define the system state by (i, j), the number of contributors and free-riders in the overlay. The state transition intensities are determined by c, the upload capacity of the contributors, and the arrival and departure intensities. The number of contributors is not limited, while the number of free-riders that receive the stream is limited by the feasibility constraint given by (1).

Figures 1 and 2 show a part of the the Markov-chains with the state transition intensities for $c = 2$ and $m = 4$.

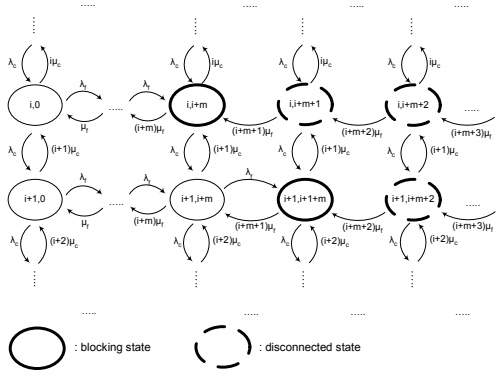


Figure 2: Markov-chain describing the overlay under the Block and Wait policy

We consider first the BD policy. The state space $\mathcal{S}$ is $\mathcal{S} := \{(i, j) : j =$

$0, 1, ..., m + (c-1)i, i = 0, 1, 2, ...\infty\}$

Arriving free-riders are blocked if the overlay does not have free upload capacity. We denote the set of blocking states as $\mathcal{B} := \{(i, m + (c-1)i) : i = 0, 1, 2, ...\infty\}$.

Once p_{ij}, the state probabilities in steady state are calculated, the performance parameters are derived as follows.

The probability that a free-rider is blocked upon arrival is

$$P(block)_{BD} = \sum_{(i,j)\in\mathcal{B}} p_{ij}. \tag{3}$$

A free-rider is dropped from the overlay if a contributor leaves when the system is in blocking state. Then, the probability that a free-rider, that has already been admitted in the overlay is later dropped is

$$P(drop)_{BD} = \frac{\sum_{(i,j)\in\mathcal{B}} i\mu_c p_{ij}}{\lambda_f(1 - P(block)_{BD})}. \tag{4}$$

The average number of nodes in the overlay is

$$\overline{N}_{BD} = \sum_{(i,j)\in\mathcal{S}} (i + j)p_{ij}. \tag{5}$$

In the case of the BW policy, the system space is infinite in both dimensions, $\mathcal{S} := \{(i, j) : i, j = 0, 1, 2, ...\infty\}$. An arriving free-rider is blocked if it arrives when the system is in states B, or when there is at least one disconnected free-rider. The set of states where there is at least one disconnected free-rider is $\mathcal{D} := \{(i, j) : j > m + (c-1)i, i = 0, 1, 2, ...\infty\}$. We can calculate the probability of blocking $P(block)_{BW}$ similarly to (3)

$$P(block)_{BW} = \sum_{(i,j)\in\mathcal{B}\cup\mathcal{D}} p_{ij}. \tag{6}$$

The average number of free-riders waiting to be reconnected to the overlay is

$$\overline{N}_{wait,BW} = \sum_{(i,j)\in\mathcal{D}} (j - (m + (c-1)i))p_{ij}. \tag{7}$$

and we can calculate the average time it takes for such a reconnection to occur using Little's form as

$$\overline{T}_{wait,BW} = \frac{\overline{N}_{wait,BW}}{\sum_{(i,j)\in\mathcal{B}\cup\mathcal{D}} i\mu_c p_{ij}}. \tag{8}$$

Finally, the average number of nodes in the overlay is

$$\overline{N}_{BW} = \sum_{(i,j)\in\mathcal{S}} (i + \min(j, m + (c-1)i))p_{ij}. \tag{9}$$

To see the effect of churn on the utilization of the overlay resources we define overlay utilization as the ratio of the average number of nodes and the maximum number of nodes a static overlay could support for given $N_c = A_c$ and α, that is,

$$U = \frac{\overline{N}}{(A_c + \min(A_f, m + (c-1)A_c)}. \quad (10)$$

To be able to calculate the state probabilities $p_{i,j}$ numerically, we limit the state space of the process. First we limit the number of contributors considered to $\{N_c^l, N_c^u\}$ as follows. We model the number of contributors in the system as an $M/M/\infty$ queue, and calculate the state probabilities p'_i. Then we select a state space symmetric to $\overline{N}_c = \frac{\lambda_c}{\mu_c}$, such that $\sum_{i=N_c^l}^{N_c^u} p'_i \geq 0.95$.

Similarly, we limit the number of free-riders considered. First we set the limits as for the contributors, then we extend the limits if necessary to include the $50(c-1)$ neighboring states of the blocking states $(i, m + (c-1)i)$.

5 Overlay Evolution in Multiple-tree Based Overlays

To see the difference between the behavior of an optimized and a more practical system, we consider multiple-tree based overlays (e.g., [6, 7]). The evolution of the transmission trees in a multiple-tree overlay is strongly correlated. This correlation makes analytical modeling complex, since the dimension of the Markov-chain describing the overlay increases linearly with the number of trees [12]. Therefore we evaluate the performance of the multiple-tree overlays via simulations.

5.1 The Multiple-tree Overlay

We assume that peer nodes are organized in t distribution trees. All the nodes are members of all t trees, and in each tree they have a different parent node from which they receive data to ensure multipath transmission. The source of the stream is the root of the trees. Since the root multiplicity is m, the root has m children in each of the trees. A contributor can have up to ct children to which it forwards data. (See Fig. 3.) We assume that nodes can have children in $d = 2$ trees if $t > 1$. This overlay architecture and the related overlay management is discussed in detail in [12].

We assume that nodes can not join a part of the trees only. It means that an arriving node is blocked if it can not get a parent in at least one tree. Similarly, a node is dropped or disconnected from all the trees if it can not reconnect after a parent departure in any of the trees.

Note, that a single transmission tree overlay with the contributors as nodes and free-riders as leaves is an optimized overlay if all contributors have the same upload

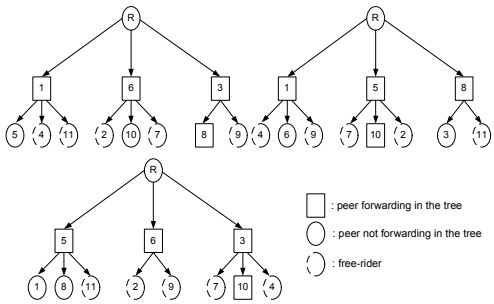


Figure 3: An overlay with $t = 3$, $m = 3$, $d = 2$ and $c = 2$

capacity. A single tree however does not provide multiple transmission path to the nodes, and therefore is not a good architecture for peer-to-peer streaming. We use single-tree simulation results to justify the analytical model.

5.2 Simulation Setup

To evaluate the multiple-tree based overlay we developed a packet level event-driven simulator. We assume that the session holding times follow an exponential distribution with mean $1/\mu = 306s$ ($\mu = \mu_c = \mu_f$), [16]. We use the arrival rate $\lambda = \lambda_c + \lambda_f$ to change A the offered load to the overlay. Under the BW policy nodes waiting to reconnect to the overlay make reconnection attempts with $0.1s$ intervals. Since we are interested in the evolution of the overlay structure we simulate only the node arrivals and departures and do not simulate the traffic flow. To obtain the results for a given offered load A and ratio of free-riders α, we start the simulation by building a static overlay. First we add $N_c = (1 - \alpha)A$ contributors then as many of αA free-riders as possible. We set $\lambda = A\mu$ and let nodes join and leave the overlay for 10000 s. The measurements are made after this warm-up period for 10000 s and the presented results are the averages of 100 simulation runs. The results have less than 5 percent margin of error at a 95 percent level of confidence.

6 Numerical Results

In this section we evaluate the service the peer-to-peer system can offer to the free-riders. We consider the probability that free riders are blocked, dropped or have to wait for reconnection. Note that contributors are never blocked, dropped or disconnected, neither in the optimized, nor in the multiple-tree overlays with flexible upload capacity allocation[12]. First we use the mathematical model to analyze the behavior of an optimized overlay, then we compare the performance of

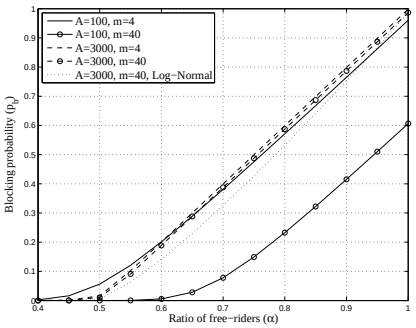


Figure 4: Block and Drop policy. Blocking probability versus α the ratio of free-riders. A=100,3000, m=4,40.

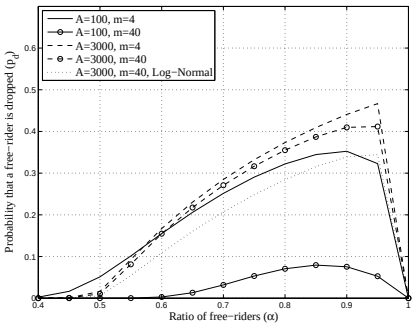


Figure 5: Block and Drop policy. Dropping probability versus α the ratio of free-riders. A=100,3000, m=4,40.

the optimized system to the one based on the less flexible multiple transmission tree structure.

Figures 4-6 show the system behavior under the BD policy as a function of α, the ratio of free-riders. Since the contributors' upload capacity is $c = 2$ the feasibility limit is around $\alpha = 0.5$, depending on the ratio of offered load and m. Both the blocking and the dropping probability increase rapidly around the feasibility limit. Below this limit the blocking and dropping values are close to zero, which means that even small systems are rather efficient. For $A = 100$ increasing the source capacity to $m = 40$ increases the feasibility limit to $\alpha = 0.7$ but the characteristics of the performance curves do not change.

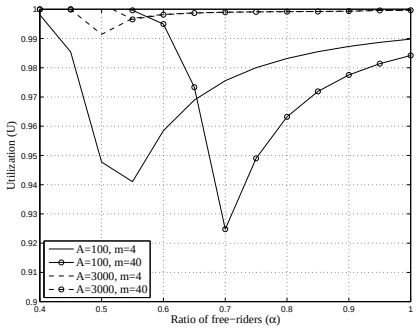


Figure 6: Block and Drop policy. Overlay utilization versus α the ratio of free-riders. A=100,3000, m=4,40.

We can also see that both the probability of blocking and the probability of dropping increase rapidly with α, that is, free-riders that were admitted are dropped with high probability at high α.

To evaluate the effect of the assumption on exponential holding times we show simulation results with log-normal holding times, $t = 1$, $A = 3000$ and $m = 40$. With log-normal holding times the blocking and dropping probabilities are slightly lower, that is, the model underestimates the performance free-riders receive.

The utilization values on Figure 6 show that in large overlays the overlay re-

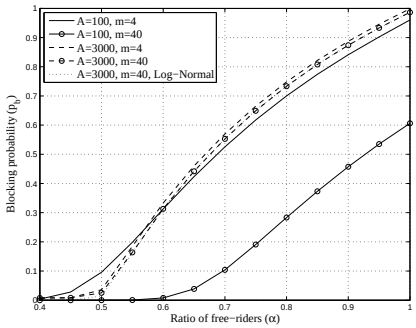


Figure 7: Block and Wait policy. Blocking probability versus α the ratio of free-riders. A=100,3000, m=4,40.

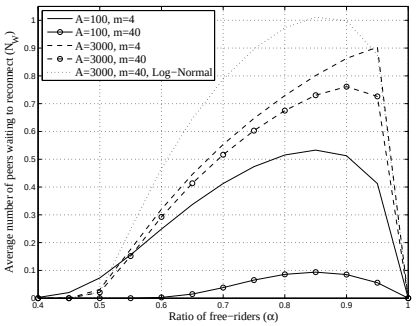


Figure 8: Block and Wait policy. Average number of peers waiting to reconnect versus α the ratio of free-riders. A=100,3000, m=4,40.

sources are stable despite churn. In small overlays the fluctuation of transmission resources decreases the utilization significantly around the feasibility limit.

Next we evaluate the BW policy. Figures 7-8 show how the blocking probability and the average number of nodes waiting for reconnection change in an optimized overlay. The simulation results with $t = 1$ and log-normal holding times fit the analytical results on the blocking probability. Simulations give higher values for the average number of nodes waiting than the model due to the deterministic time intervals of reconnection attempts. If we decrease the reconnection interval the difference decreases as well. Figure 9 shows the average time nodes have to wait to reconnect, normalized by the average holding time μ_f. The normalized reconnection time is very low for all cases.

Comparing the BD and BW policies, we can see that under the BW policy free-riders are blocked with higher probability, while even at high ratio of free-riders there are very few nodes waiting for reconnection and the waiting time to reconnect is very low. From this we can conclude that under the BW policy free-riders that are admitted will receive the stream with little disturbances. As figure 10 shows the utilization of the overlay resources are very similar under the BD and the BW policies, despite the different characteristics of blocking.

Finally, we compare the performance of the optimized overlays and of overlays based on multiple-tree structure. Figure 11 compares the utilization of single tree ($t = 1$) and multiple-tree overlays for $t = 8$ under the BD policy. As t increases, the probability that free-riders are dropped increases as well, since nodes that can not reconnect to even one tree are dropped. This generates free capacity in the overlay, which in turn leads to decreasing blocking probability for increasing t. As a result, the utilization of overlay resources is not sensitive to the number of trees in

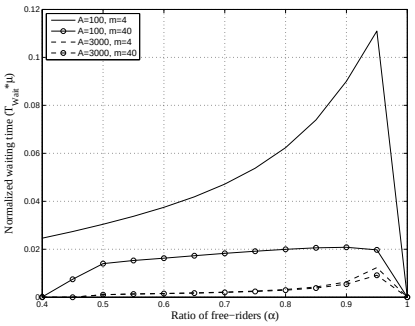


Figure 9: Block and Wait policy. Average waiting time versus α the ratio of free-riders. A=100,3000, m=4,40.

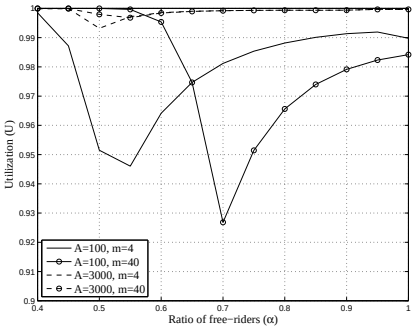


Figure 10: Block and Wait policy. Overlay utilization versus α the ratio of free-riders. A=100,3000, m=4,40.

large overlays ($A = 3000$). In small overlays ($A = 100$) there can be a loss of 20%, increasing with t. Figure 12 shows the overlay utilization for the same scenarios, but for the BW policy. Comparing figures 11 and 12, the BW policy seems to be more efficient, but its efficiency depends on the interval of reconnection attempts.

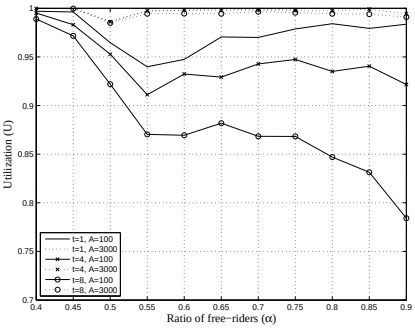


Figure 11: Multiple-tree overlay, Block and Drop policy. The effect of multiple trees on the overlay utilization. Simulation results.

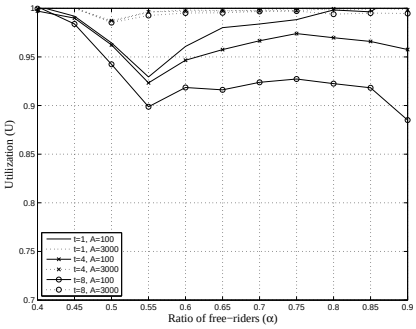


Figure 12: Multiple-tree overlay, Block and Wait policy. The effect of multiple trees on the overlay utilization. Simulation results.

7 Conclusion

In this paper we evaluated the performance goodwill free-riders receive in a peer-to-peer streaming overlay. We compared two policies to limit the number of free-riders in the system, the Block and Drop and the Block and Wait policies. We showed analytically that under the Block and Drop policy both the blocking and the dropping probabilities can be high, which means that the free-riders already admitted to the system are often dropped. Under the BW policy, however, the number of free-riders waiting to be reconnected is very low for all parameter settings, which suggests that once admitted, free-riders receive the stream without interruption with high probability. This feature makes the BW policy a good candidate to control free-riders.

We compared the performance of optimized overlays and overlays based on rigid multiple-tree structures. We showed that in a multiple-tree overlay the blocking and dropping probabilities and the average number of free-riders waiting depend on the number of distribution trees, but the resulting overlay utilization approaches the one of the optimized overlay.

References

[1] G. Tan and S.A. Jarvis. A payment-based incentive and service differentiation mechanisms for peer-to-peer streaming broadcast. In *Proceedings of IWQoS*, 2006.

[2] Y. Sung, M. Bishop, and S. Rao. Enabling contribution awareness in an overlay broadcasting system. In *ACM Sigcomm*, 2006.

[3] X. Hei, Ch. Liang, Y.Liu, and K.W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 2007.

[4] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for p2p streaming systems. In *IEEE INFOCOM*, 2007.

[5] V. Fodor and Gy. Dán. Resilience in live peer-to-peer streaming. *IEEE Communications Magazine*, 45(6), 2007.

[6] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: High-bandwidth multicast in a cooperative environment. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[7] Venkata N. Padmanabhan, Helen J. Wang, and Philip A. Chou. Resilient peer-to-peer streaming. In *Proceedings of the 11th IEEE International Conference on Network Protocols*, 2003.

[8] X. Zhang, J. Liu, B. Li, and T-S. P. Yum. Coolstreaming/donet: a data driven overlay network for efficient live media streaming. In *Proceedings of IEEE INFOCOM*, 2005.

[9] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *Proceedings of IEEE INFOCOM*, 2007.

[10] R. Rejaie N. Magharei. Prime: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of IEEE INFOCOM*, 2007.

[11] T. Small, B. Liang, and B. Li. Scaling laws and tradeoffs in peer-to-peer live multimedia streaming. In *Proceedings of ACM Multimedia*, 2006.

[12] Gy. Dán, V. Fodor, and I. Chatzidrossos. On the performance of multiple-tree-based peer-to-peer live streaming. In *Proceedings of IEEE INFOCOM*, 2007.

[13] Gy. Dán and V. Fodor. An analytical study of low delay multi-tree-based overlay multicast. In *Proceedings of ACM Peer-to-peer Streaming and IPTV Workshop*, 2007.

[14] L. Massoulie, A. Twigg, Ch. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithms. In *Proceedings of IEEE INFOCOM*, 2007.

[15] D. Carra, R. Lo Cigno, and E.W. Biersack. Graph based modeling of p2p streaming systems. In *Proceedings of IFIP Networking*, 2007.

[16] E. Veloso V. Almeida W. Meira A. Bestavros and S. Jin. A hierarchical characterization of a live streaming media workload. In *Proceedings of ACM IMC*, 2002.

[17] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the internet. In *Proceedings of ACM IMC*, 2004.

Paper D

Server Guaranteed Cap: An incentive mechanism for maximizing streaming quality in heterogeneous overlays

Ilias Chatzidrossos, György Dán and Viktória Fodor.
In Proc. of IFIP Networking, Chennai, India, May 2010.

Server Guaranteed Cap: An incentive mechanism for maximizing streaming quality in heterogeneous overlays

Ilias Chatzidrossos, György Dán, Viktória Fodor
Laboratory for Communication Networks
School of Electrical Engineering
KTH, Royal Institute of Technology

Abstract

We address the problem of maximizing the social welfare in a peer-to-peer streaming overlay given a fixed amount of server upload capacity. We show that peers' selfish behavior leads to an equilibrium that is suboptimal in terms of social welfare, because selfish peers are interested in forming clusters and exchanging data among themselves. In order to increase the social welfare we propose a novel incentive mechanism, Server Guaranteed Cap (SGC), that uses the server capacity as an incentive for high contributing peers to upload to low contributing ones. We prove that SGC is individually rational and incentive compatible. We also show that under very general conditions, there exists exactly one server capacity allocation that maximizes the social welfare under SGC, hence simple gradient based method can be used to find the optimal allocation.

1 Introduction

The goal of peer-to-peer (p2p) streaming systems is to achieve the maximum possible streaming quality using the upload capacities of the peers and the available server upload capacity. In general, the achievable streaming quality depends heavily on the aggregate upload capacity of the peers [1]. Hence, a key problem of p2p streaming systems is how to give incentives to selfish peers to contribute with all their upload capacity. Numerous schemes were proposed to solve this problem (e.g., [2, 3]). These schemes relate peers' contribution with the streaming quality they receive: the more a peer contributes, the better streaming quality it can potentially receive. The correlation of peer contribution to the quality it receives is based on the assumption that all peers are capable of contributing but refrain from doing so.

Nevertheless, peers might be unable to have a substantial contribution with respect to the stream rate because of their last-mile connection technology. Most DSL and cable Internet connections are asymmetric, hence peers may have sufficient capacity to, e.g., download high definition video but insufficient for forwarding it. Similarly, in high-speed mobile technologies, such as High Speed Downlink Packet Access (HSDPA), the download rates are an order of magnitude higher than the upload rates [4]. Peers using assymetric access technologies would receive poor quality under incentive schemes that offer a quality proportional to the level of peer contribution.

Furthermore, using such incentive schemes high contributing peers maximize their streaming quality if they prioritize other high contributing peers when uploading data. As a consequence, peers with similar contribution levels form clusters and exchange data primarily among themselves. While high contributing peers can achieve excellent streaming quality this way, the quality experienced by low contributing peers is low, and the average streaming quality in the p2p system is suboptimal.

In order to increase the average streaming quality in the system, we propose a mechanism that gives incentives to high contributing peers to upload to low contributing ones. The mechanism relies on reserving a portion of the server capacity and providing it as a safety resource for high contributing peers who meet certain criteria. We show that high contributing peers gain by following the rules set by the incentive mechanism, and they fare best when they follow the rules truthfully. We also show that due to some basic properties of p2p streaming systems our mechanism can easily be used to maximize the streaming quality.

The rest of the paper is organized as follows. In Section 2, we motivate our work by studying the effect of selfish peer behavior in a push-based p2p streaming overlay. In Section 3, we describe our incentive mechanism and provide analytical results. We show performance results in Section 4. In Section 5 we discuss previous works on incentives in peer-to-peer streaming systems. Finally, Section 6 concludes our paper.

2 Motivation

In order to understand the effects of selfish peer behavior on the performance of p2p streaming systems, in the following we study a mesh-based p2p streaming system. Mesh-based systems have received significant attention in the research community [5, 6, 7, 8, 9], and are underlying the majority of commercial streaming systems (e.g., [10], [11]).

2.1 Case study: a mesh-based p2p streaming system

The streaming system we use as an example was proposed in [8] and was subsequently analyzed in [9, 12]. The system consists of a server and N peers. The

upload capacity of the server is m_t times the stream rate. For simplicity we consider two types of peers: peers with high upload capacity, called contributors, and peers without upload capacity, called non-contributors. The upload capacity of the contributors is c times the stream rate, while that of non-contributors is zero. We denote by α the ratio of non-contributors in the overlay.

Each peer is connected to d other peers, called its neighbors. The server is neighbor to all peers. Every peer maintains a playout buffer of recent packets, and exchanges information about its playout buffer contents with its neighbors periodically, via so called buffer maps. The server sends a copy of every packet to m_t randomly chosen peers. The peers then distribute the packets among each other according to a forwarding algorithm. The algorithm takes as input the information about packet availability in neighboring peers (known from the buffer maps) and produces a forwarding decision, consisting of a neighbor and a packet sequence number. In this work we consider the Random Packet - Random Peer ($RpRp$) forwarding algorithm. This algorithm has been shown to have a good playback continuity - playback delay tradeoff ([8, 12]). According to this algorithm, a sending peer first chooses randomly a neighbor that is missing at least one packet that itself has and it sends to that neighbor a random packet that it is missing. Peers play out data B time after they were generated by the server, and we refer to this as the playback delay.

2.2 Playout probability, individual utility and social welfare

The performance of p2p streaming systems is usually measured in terms of the playout probabilities of the peers, i.e., the probability p_i that peer i receives packets upon their playout deadlines [8, 12, 9]. The impact of the playout probability p_i on the peers' satisfaction is, however, typically influenced by the loss resilience of the audiovisual encoding. To allow for a wide range of encodings, we use utility functions to map the playout probability to user satisfaction. Formally, the utility function is a mapping $u : [0, 1] \rightarrow [0, 1]$. We consider three kinds of utility functions.
Linear function: Utility function of the form $y = a \cdot p_i + b$. An improvement in the playout probability yields the same increase in utility regardless of the already achieved playout probability.
Concave function: Utility is a concave function of the playout probability, that is, the marginal utility is a non-increasing function of the playout probability.
Step function: There is an instantaneous transition from a zero utility to a utility of a unit upon reaching a threshold p_i^*. The peer is only satisfied above the threshold playout probability.

We measure the aggregate utility of the peers, called the social welfare, using the utilitarian welfare model. In the utilitarian welfare model the social welfare is the sum of the utilities, which is equivalent to the average peer utility

$$SWF = (1 - \alpha) \cdot u(p_c) + \alpha \cdot u(p_{nc}), \tag{1}$$

where p_c and p_{nc} denote the playout probability of contributors and non-contributors respectively.

2.3 Modelling peer behavior

To study the impact of peer cooperation in the overlay, we introduce the notion of generosity factor, which we denote by β. This parameter shows how generous a peer is towards its non-contributing neighbors. The generosity factor takes values in the interval $[0, 1]$. When $\beta = 1$, the peers are completely generous and they upload to their neighbors regardless of whether they, on their turn, are uploading or not. When $\beta = 0$, peers are not generous at all, or equivalently completely selfish, and will only upload to peers that upload as well. Values of the generosity level in the set $(0, 1)$ correspond to intermediate generosity levels.

The generosity level affects the playout probabilities of the contributing and non-contributing peers. Since the major source of capacity for the non-contributors are the contributors, the generosity level of contributors determines their playout probability. When $\beta = 1$, then contributors share equally their capacity between their contributing and non-contributing neighbors. As β decreases, capacity is subtracted from the non-contributors and added to the contributors. This means that the playout probability of the contributors increases, while that of non-contributors decreases.

2.4 The effects of selfish behavior

In the following we show the effects of selfish behavior on the social welfare. The numerical results we show were obtained using an analytical model and via simulations. The analytical model is an extension of our previous work [12], where we developed a model of the playout probability in a push-based system with homogeneous peer upload capacities. We extended the model to incorporate two types of peers, contributors and non-contributors. The extended model can be found in [13]. The simulation results were obtained using the packet-level event-driven simulator used in [12]. In the simulations, nodes join the overlay at the beginning of the simulation, and are organized into a random, d-regular graph. After the overlay with N peers is built, the data distribution starts according to the forwarding algorithm described in Section 2.1. The algorithm is executed in time slots in a way that contributors with capacity c make c forwarding decisions per slot. All results presented in the following refer to an overlay of $N = 500$ peers, where each peer is connected to $d = 30$ neighbors and contributors have an upload capacity of $c = 2$ times the stream bitrate.

Fig. 1a and 1b show the effect of the generosity factor on the playout probability of the contributors and the non-contributors obtained using the model and simulations. The figures also show the average playout probability in the overlay (dotted lines). The share of non-contributors is $\alpha = 0.5$. Fig. 1a shows a system

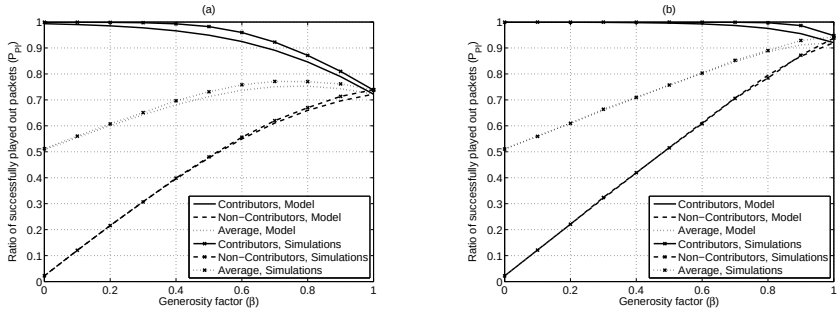


Figure 1: Ratio of successfully played out packets vs generosity factor β for playback delay of (a) $B = 10$ and (b) $B = 40$. Analytical and simulation results.

where the playback delay is small. Clearly, contributors maximize their playout probabilities for $\beta = 0$, but the average playout probability is suboptimal in this case. Surprisingly, the average playout probability is suboptimal for $\beta = 1$ as well. This shows that when β is small, the upload capacity is better utilized when allocated to non-contributors, but for large β values it is better utilized by contributors. Nevertheless, for a larger playback delay (Fig. 1b) the average playout probability is maximized for $\beta = 1$. This is because for such a large playback delay the upload capacity is better utilized when used to feed the non-contributors for all β. From these two figures we draw the conclusion that average playout probability depends on the contributors' generosity, but $\beta = 1$ is not necessarily optimal.

The performance degradation when peers act selfishly is a general characteristics of mesh-based p2p streaming systems. The inefficiencies in data delivery are due to the lack of coordination between peers [14], which prohibits seamless streaming. In push-based algorithms [8, 12] the lack of coordination leads to duplicate packet receptions at peers, i.e., a peer receives the same data from more than one of its neighbors. Fig. 2 supports that the decrease in the average playout probability is actually due to the clustering of contributors, which leads to the increase in the ratio of duplicate transmissions. In the case of pull-based systems the lack of coordination leads to multiple requests for data at sending peers, also a source of inefficient bandwidth use. While coordination can mitigate the inefficiency [14], it leads to higher delay in delivering the

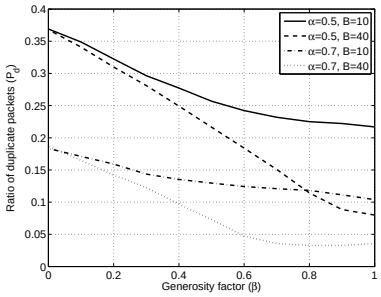


Figure 2: Ratio of duplicate transmissions vs generosity factor (β) for $B = 10$ and $B = 40$. Simulations.

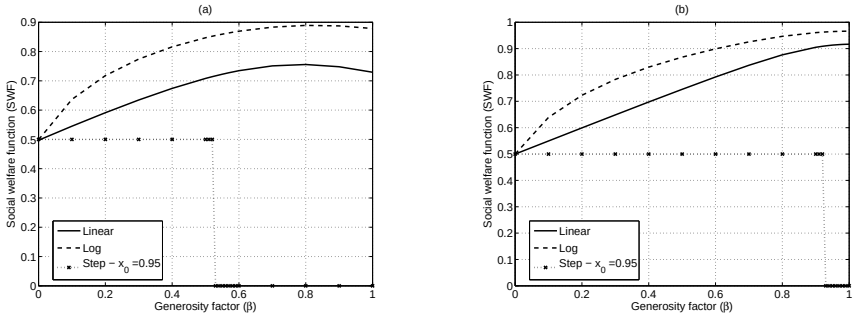


Figure 3: Social welfare vs. generosity factor β for playback delays of (a) $B = 10$ and (b) $B = 40$. Overlay with $\alpha = 0.5$. Analytical results.

data, which in turn leads to lower playout probabilities for any fixed playback delay. Based on these findings, we argue that performance degradation due to peer clustering is not restricted to push systems only, but is intrinsic to uncoordinated p2p dissemination algorithms.

Next, we proceed with our utility based analysis of the overlay. For linear utility function we use $u(p_i) = p_i$, so the utility curve coincides with the curve presented in Fig. 1a and 1b. For concave utility we use a logarithmic function, $u(p_i) = log_{10}(1 + 9p_i)$. For the step function we set the threshold to $p_i^* = 0.95$. Our conclusions do not depend on the the particular values of the parameters and the choice of the logarithmic function.

Fig. 3a and 3b show the social welfare versus the generosity factor for the three kinds of utility functions and for playback delays of $B = 10$ and $B = 40$ respectively. In the case of small playback delay ($B = 10$) the social welfare for the linear and the concave utility functions attains its maximum for $\beta < 1$. For the step function the social welfare equals 0 for high values of β, when contributors are not able to receive at least with probability $p_i^* = 0.95$. As β decreases, there is a transition in utility, but the contributors do not gain anything by becoming more selfish after the transition, and the social welfare remains constant. In the case of large playback delay ($B = 40$), we see that the social welfare for linear and concave utility functions attains its maximum for $\beta = 1$. For the step function we observe a similar transition of the social welfare as for $B = 10$, but at a higher value of the generosity factor β. The transition occurs where the contributors achieve a playout probability of $p_i^* = 0.95$. To understand the importance of the threshold value, let us consider $p_i^* = 0.8$. It is easy to see based on Fig. 1b that in this case the social welfare would be maximal for $\beta \geq 0.8$, as both contributors and non-contributors achieve playout probabilities above the threshold. To summarize, we draw two conclusions from these figures. First, for the linear and concave utility functions the value of β that maximizes the social welfare is a function of the playback delay, but in general

$\beta = 0$ is far from optimal. Second, for the step function the threshold value p_i^* plays an important role in whether $\beta = 0$ is optimal.

3 The SGC incentive mechanism

Our work is motivated by the observation that the peers' selfish behavior leads to a loss of social welfare. In our solution, we exploit the inability of contributors to achieve the maximum playout probability by being selfish and offer them seamless streaming if they increase their generosity, that is if they serve non-contributors as well. In the following, we describe our incentive mechanism, called Server Guaranteed Cap (SGC).

Under SGC, there are two types of data delivery: *p2p dissemination* and *direct delivery* from the server. Fresh data is distributed in the overlay using *p2p dissemination*: the peers forward the data according to some forwarding algorithm. Contributors can also request data that they miss *directly from the server* if they do not exceed a threshold playout probability of T_p via p2p dissemination. In our scheme the server ensures that by combining p2p dissemination and direct delivery the contributors possess all data with probability 1. In order to be able to serve the requests for direct delivery, the server reserves m_r of its total upload capacity m_t for direct delivery. m_r has to be large enough to cap the gap between the threshold probability T_p and 1. Given the number of contributors in the overlay and the reserved capacity m_r, the server can calculate the threshold value of the playout probability below which it would not be able to cap all contributors

$$T_p = 1 - \frac{m_r}{(1-\alpha) \cdot N} \tag{2}$$

The server advertises the threshold value T_p as the maximum playout probability that contributors should attain through p2p dissemination. Peers report their playout probabilities p_i achieved via p2p dissemination. Based on the reports, the server knows which are the contributors with $p_i \leq T_p$, that is, which contributors are entitled for direct delivery.

3.1 Properties of SGC

In the following we show two important properties of the proposed mechanism: ex-post individual rationality and incentive compatibility. Ex-post individual rationality means that a contributing peer does *not* achieve *inferior* performance by following the rules of the mechanism irrespective of the playout probability it would achieve without following the mechanism.

Proposition 1. *The SGC mechanism is ex-post individually rational.*

Proof. Consider that the server advertises a threshold probability of T_p. All contributors that receive up to $p_i \leq T_p$ via p2p dissemination are entitled to pull the

remaining $1 - T_p$ directly from the server. Hence a peer with $p_i = T_p$ receives data with probability $P_i = p_i + (1 - T_p) = 1$, which is at least as much as it would achieve by not following the rules of the mechanism. □ □

Since *SGC* relies on peers reporting their playout probabilities to the server, it is important that peers do not have an incentive to mis-report their playout probabilities. In the following we show that *SGC* satifies this criterion, i.e., it is incentive compatible.

Proposition 2. *The SGC mechanism is incentive compatible.*

Proof. Let us denote the playout probability of peer i by p_i and the probability it reports by $\overline{p}_i$. T_p is the threshold probability that the contributors must not exceed in order to be directly served by the server, and m_r is the corresponding reserved capacity at the server. Contributors can thus receive $m_r/(1-\alpha)N = 1 - T_p$ share of the stream from the server directly if they report $\overline{p}_i \leq T_p$. Consequently, if peer i achieves $p_i \leq T_p$ and reports it truthfully ($\overline{p}_i = p_i$), it receives data with probability $P_i = min(1, p_i + (1 - T_p))$. If $p_i > T_p$ and peer i reports truthfully, it receives with probability $P_i = p_i$.

Clearly, peer i can not benefit from over-reporting its playout probability, so we only have to show that it has no incentive for under-reporting it either. In order to show this we distinguish between three cases.

- $\overline{p} < p \leq T_p$: the playout probability that the peer will finally receive will be $P_i = min(1, p + 1 - T_p) \leq 1$, which is the same that it would receive if it were telling the truth
- $\overline{p} \leq T_p < p$: the playout probability that the peer will finally receive will be $P_i = min(1, p + 1 - T_p) = 1$. The peer could achieve the same by having $p = T_p$ and reporting $\overline{p} = p$.
- $T_p < \overline{p} < p$: the peer is not entitled to direct delivery, so $P_i = p_i$. □

□

We note that apart from not gaining, peer i can actually lose by declaring that it does not have packets it already has: it can reduce the probability of receiving packets from its neighbors that it does not possess in reality. We did not have to account for this to prove incentive compatibility, however.

3.2 Optimal server capacity allocation

A key question for the implementation of the mechanism is how to determine the advertised probability threshold T_p, that is, how to find the reserved capacity m_r. Since the server capacity is fixed, the choice of m_r affects the server capacity available for the p2p dissemination, and hence the efficiency of the data delivery through

p2p dissemination. To study the effect of server resource allocation on the playout probability of the peers, we express the success of a forwarding algorithm in delivering the stream as a function of the overlay size and the capacity allocated to the peers. We define the mapping $f : (\mathbb{N}, \mathbb{R}) \rightarrow [0, 1]$ of number of peers in an overlay and the aggregate capacity allocated to them, to the average playout probability of the peers.

In the following we show that for a wide class of p2p streaming systems there is a unique value of m_r that maximizes the social welfare. This class of p2p streaming systems is characterized by the fact that the marginal gain of increasing the upload capacity in the system is non-increasing.

Definition 1. *A p2p streaming system is called efficient if the playout probability of the peers is a concave function of the overlay capacity.*

We only consider systems where the efficiency of the forwarding algorithm does not depend on the overlay size for a given ratio of peers over total upload capacity, that is for systems where it holds that $f(k \cdot N, k \cdot C) = f(N, C), \forall k \in \mathbb{R}$. Given that, we formulate the following proposition.

Proposition 3. *The construction of an efficient p2p streaming system is always feasible regardless of the characteristics of the forwarding algorithm used.*

Proof. Let us consider a system with N peers and total upload capacity C. Suppose that f is strictly convex on an interval in its domain. Formally, there exist upload capacity values C_1, C_2, with $C_1 < C_2$, for which it holds

$$\lambda f(N, C_1) + (1 - \lambda) f(N, C_2) > f(N, \lambda C_1 + (1 - \lambda) C_2), \forall \lambda \in (0, 1) \tag{3}$$

We split the overlay into two partitions, one with size λN and server capacity λC_1 and the other with $(1 - \lambda)N$ peers and $(1 - \lambda)C_2$ server capacity. For the two overlays we have that

$$f(\lambda N, \lambda C_1) = f(N, C_1) \tag{4}$$

$$f((1 - \lambda)N, (1 - \lambda)C_2) = f(N, C_2) \tag{5}$$

By combining (4) and (5) we get $f(N, \lambda C_1 + (1 - \lambda)C_2) = \lambda f(N, C_1) + (1 - \lambda) f(N, C_2)$, which contradicts (3). Hence, by splitting the overlay we can create an efficient p2p streaming system. □

For a given server capacity, m_t, SGC requires that the server caps the gap between the playout probability p_i achieved via p2p dissemination and 1. Therefore, the value of m_r should be such that contributors can achieve T_p for some $\beta \in [0, 1]$. The feasible range for the implementation of SGC is then defined as $\mathbb{M}_r = \{m_r \in (0, m_t) : m_r \geq (1 - f\big((1 - \alpha) \cdot N, C_t\big)) \cdot (1 - \alpha) \cdot N\}$, where C_t is the total capacity of the contributors and the server capacity used for the p2p dissemination.

Proposition 4. *For an efficient system, a feasible range of server upload capacities $\mathbb{M}_r$ and a concave utility function, the social welfare is a concave function of the reserved server capacity m_r.*

Proof. The social welfare of the system is given as

$$SWF = (1-\alpha)\cdot u\Big(f((1-\alpha)\cdot N, C_c) + \frac{m_r}{(1-\alpha)\cdot N}\Big) + \alpha \cdot u\Big(f(\alpha\cdot N, C_{nc})\Big) \quad (6)$$

where C_c and C_{nc} are the upload capacities allocated to contributors and to non-contributors respectively (as a function of β), and $C_t = C_c + C_{nc}$. Under the *SGC* mechanism, the contributors receive the stream with probability 1, so the above equation becomes

$$SWF = (1-\alpha)\cdot u(1) + \alpha \cdot u\Big(f(\alpha\cdot N, C_{nc})\Big). \quad (7)$$

In the following we show that C_{nc} is a concave function of m_r. Since our system is efficient the playout probability p_i is concave with respect to C_c, or equivalently C_c is convex in p_i=T_p. Since $C_{nc} = C_t - C_c$, C_{nc} is concave in $1 - T_p$, which on its turn is linear in m_r. Therefore, C_{nc} is a concave function with respect to m_r. Consequently the composite function $f(C_{nc})$ is concave as well with respect to m_r, as a composition of non-decreasing concave functions [15]. For the same reason $u \circ f$ is concave with respect to m_r, which proves the proposition. □

A consequence of *Proposition 4* is that the social welfare function SWF has exactly one, global, maximum on $\mathbb{M}_r$. Hence, the server can discover the optimal amount of reserved capacity m_r by using a gradient based method starting from any $m_r \in \mathbb{M}_r$.

4 Numerical results

In the following we show numerical results that quantify the gain of the proposed incentive mechanism. The social welfare with respect to the reserved capacity by the server is shown in Fig. 4. The total capacity of the server is $m_t = 11$. We can see that the feasible region of *SGC* depends on the playback delay for the system. For B=10, it holds that $m_r \in [2, 9]$, while for larger playback delays $m_r \in [1, 10]$. The increase of m_r triggers two contradicting effects. On one side, it increases the playout probability of contributors through the direct delivery. On the other side, it decreases the efficiency of the p2p dissemination phase, since the amount of server capacity dedicated to that type of dissemination is decreased. The social optimum is at the allocation where the rate of decrease of the efficiency of p2p dissemination becomes equal to that of the increase achieved through the direct delivery.

Finally, we note that even using *SGC* there is a *loss of social welfare* compared to the *hypothetical case* when β is optimized by generous peers to maximize the

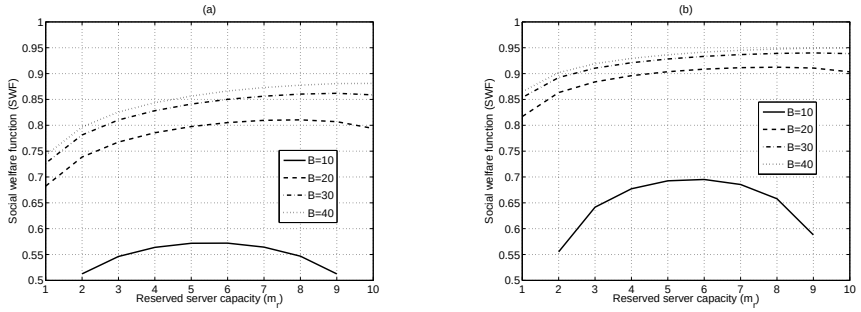


Figure 4: Social welfare versus reserved server capacity for different playback delays. Linear (a) and logarithmic (b) utility functions. Overlay with $N = 500$, $d = 30$ and $m = 11$.

social welfare. We can observe this loss by comparing the maximum social welfare obtained in Figs. 4 to that in Fig. 3 (for $B = 10$ and $B = 40$). This loss of social welfare is the *social cost of the selfishness of peers*.

5 Related work

A large number of incentive mechanisms was proposed in recent years to solve the problem of free-riding in p2p streaming systems. These mechanisms are either based on pairwise incentives or on global incentives.

Pairwise incentive schemes were inspired by the tit-for-tat mechanism used in the BitTorrent protocol [3, 16]. However, tit-for-tat, as used in BitTorrent, was shown not to work well in live streaming with random neighbor selection [3, 16]. The authors in [16] proposed an incentive mechanism for neighbor selection based on playback lag and latency among peers, achieving thus better pairwise performance. In [17], the video was encoded in layers and supplier peers favored neighbors that uploaded back to them, achieving thus service differentiation as well as robustness against free-riders.

Global incentive schemes take into account the total contribution of a peer to its neighbors. In [2], a rank-based tournament was proposed, where peers are ranked according to their total upload contribution and each peer can choose as neighbor any peer that is below itself in the ranked list. Thus, peers that have high contribution have also higher flexibility in selecting their neighbors. In [18], the authors proposed a payment-based incentive mechanism, where peers earn points by uploading to other peers. The supplier peer selection is performed through first price auctions, that is, the supplier chooses to serve the peer that offers her the most points. This means that peers that contribute more, accumulate more points and can make high bids, increasing thus their probability of winning an auction for a parent.

All the aforementioned incentive mechanisms assume that peers are always capable of contributing but, due to selfishness, refrain from doing so. We, on the contrary, consider peers that are unable to contribute because of their access technologies. Associating streaming quality with contribution unnecessarily punishes these weak peers. Therefore our goal is to maximize the social welfare in the system, by having the high contributing peers upload to low contributing peers as well. In this aspect, our work is closely related to [19], where a taxation scheme was proposed, based on which high contributing peers subsidize low contributing ones so that the social welfare is maximized. However, in contrast to [19], where it is assumed that peers voluntarily obey to the taxation scheme and they can only react to it by tuning their contribution level, we prove that our mechanism is individually rational and incentive compatible. To the best of our knowledge our incentive scheme is unique in these two important aspects.

6 Conclusion

In this paper we addressed the issue of maximizing the social welfare in a p2p streaming system through an incentive mechanism. We considered a system consisting of contributing and non-contributing peers and studied the playout probability for the two groups of peers. We showed that when contributing peers are selfish the system operates in a state that is suboptimal in terms of social welfare. We proposed an incentive mechanism to maximize the social welfare, which uses the server's capacity as an incentive for contributors to upload to non-contributing peers as well. We proved that our mechanism is both individually rational and incentive compatible. We introduced the notiona of efficient p2p systems and proved that for any efficient system there exists exactly one server resource allocation that maximizes the social welfare. An extension of our scheme to several classes of contribution levels will be subject of our future work.

References

[1] Kumar, R., Liu, Y., Ross, K.: Stochastic fluid theory for p2p streaming systems. In: IEEE INFOCOM. (2007)

[2] Habib, A., Chuang, J.: Service differentiated peer selection: An incentive mechanism for peer-to-peer media streaming. IEEE Transactions on Multimedia **8** (June 2009) 610–621

[3] Silverston, T., Fourmaux, O., Crowcroft, J.: Towards an incentive mechanism for peer-to-peer multimedia live streaming systems. In: IEEE International Conference on Peer-to-Peer Computing. (2008)

[4] Chahed, T., Altman, E., Elayoubi, S.: Joint uplink and downlink admission control to both streaming and elastic flows in CDMA/HSDPA systems. Performance Evaluation **65** (November 2008) 869–882

[5] N. Magharei, R.R.: Prime: Peer-to-peer receiver-driven mesh-based streaming. In: Proc. of IEEE INFOCOM. (2007)

[6] M. Zhang, L. Zhao, Y.T., Luo, J.G., Yang, S.Q.: Large-scale live media streaming over peer-to-peer networks through the global internet. In: Proc. ACM Workshop on Advances in peer-to-peer multimedia streaming (P2PMMS). (2005)

[7] A. Vlavianos, M. Iliofotou, M.F.: Bitos: Enhancing BitTorrent for supporting streaming applications. In: Proc. of IEEE INFOCOM. (2006)

[8] Bonald, T., Massoulié, L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: Optimal performance trade-offs. In: Proc. of ACM SIGMETRICS. (2008)

[9] Liang, C., Guo, Y., Liu, Y.: Investigating the scheduling sensitivity of p2p video streaming: An experimental study. IEEE Transactions on Multimedia **11** (April 2009) 348–360

[10] PPLive: http://www.pplive.com/en/about.html.

[11] UUSee: http://www.uusee.com

[12] Chatzidrossos, I., Dán, G., Fodor, V.: Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates. P2P Networking and Applications (May 2009)

[13] Chatzidrossos, I., Dán, G., Fodor, V.: Server Guaranteed Cap: An incentive mechanism for maximizing streaming quality in heterogeneous overlays. Technical Report TRITA-EE 2009:059, KTH, Royal Institue of Technology (Dec 2009)

[14] Picconi, F., Massoulié, L.: Is there a future for mesh-based live video streaming? In: IEEE International Conference on Peer-to-Peer Computing. (2008)

[15] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)

[16] Pianese, F., Perino, D.: Resource and locality awareness in an incentive-based P2P live streaming system. In: Proc. of the Workshop on Peer-to-peer Streaming and IP-TV. (2007)

[17] Liu, Z., Shen, Y., Panwar, S.S., Ross, K.W., Wang, Y.: Using layered video to provide incentives in p2p live streaming. In: Proc. of Workshop on Peer-to-Peer Streaming and IP-TV. (2007)

[18] Tan, G., Jarvis, S.A.: A payment-based incentive and service differentiation mechanism for peer-to-peer streaming broadcast. In: Proc. of 14th International Workshop on Quality of Service (IWQoS). (2006)

[19] Chu, Y., Chuang, J., Zhang, H.: A case for taxation in peer-to-peer streaming broadcast. In: Proc. of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems. (2004)

Appendix: Data Propagation Model

We denote by P_i^j the probability that an arbitrary peer has packet j at the end of time-slot i. Each peer buffers packets for a number of slots before it plays them out in order to increase the probability of uninterrupted playback. The buffer size is the playback delay (B) of the peer. Thus, a packet that is generated at time-slot j will be played out at time slot $j+B$. In order for a packet to be successfully played out, it has to be at the buffer of the peer by the end of the time-slot that precedes its playout slot. Namely the playout probability of packet j is expressed as P_{j+B-1}^j.

The forwarding of packets by contributors depends on the generosity factor β, which shows how generous contributors are towards their non-contributing neighbors. We denote by p, the probability that at any given slot a contributor will choose to forward a packet to one of its non-contributing neighbors. Then, the generosity factor can be defined as $\beta = p/\alpha$. Intuitively, the generosity factor can be considered as the fraction of time that a contributing peer is forwarding to non-contributing neighbors over the ratio of its non-contributing neighbors.

A peer can have packet j at the end of time-slot j only if it received it directly from the server. At the end of any time-slot i, with $i > j$, a peer can have packet j if it already had it at the end of time-slot $i-1$ or if it received it during time-slot i from any of its contributing neighbors. This is expressed by the following equation:

$$P_i^j = \begin{cases} 0 & i < j \\ \frac{m}{N} & i = j \\ P_{i-1}^j + (1 - P_{i-1}^j)(1 - (1 - \pi_i^j)^{D^c}) & i > j. \end{cases} \quad (8)$$

where D^c is the number of contributing neighbors of a peer and π_i^j is the probability that packet j was sent by a contributing neighbor during time-slot i.

Since the overlay is a random regular graph, the average number of contributing neighbors of contributors and non-contributors is the same. However, for values of the generosity factor β of less than 1, the probability π_i^j differs between contributors and non-contributors, resulting in a different probability P_i^j. Therefore, in order to

describe the data propagation in a heterogeneous overlay, we need two equations of the form of (8), one for the contributors and another for the non-contributors. We denote by ξ_i^j and ϕ_i^j the probability P_i^j for a contributor and a non-contributor respectively.

Next, we proceed with the calculation of the probability π_i^j. For that, we consider an arbitrary peer r, who does not have packet j at the beginning of time-slot i, and one of its contributing neighbors, s. Since contributors may forward up to c packets per slot, we divide the slot in c sequential selections of a pair (neighbor,packet). For the sake of simplicity we assume that s does not keep memory of the pairs that it has chosen in the previous selections within a time-slot. That means, we allow s to choose the same pair of neighbor and packet more than once in a time-slot. The probability that r will receive packet j from s, is the probability that in one of these selections s will choose the pair (r,j). Let us denote by $\ell(i)$ the lowest packet sequence number that peers would potentially forward in time slot i. Since in slot i a peer would not forward the packet that is to be played out in that slot, $\ell(i) = max\{0, i - B + 1\}$. The highest packet sequence number that any peer can forward in slot i is $i - 1$, because only the source has packet i in slot i.
To calculate π_i^j, we define the events:

$A := \{s \text{ has packet } j\}$,
$C := \{s \text{ chooses to send a packet to } r\}$,
$D := \{s \text{ chooses to send packet } j \text{ to } r\}$,
$E := \{s \text{ sends to contributor during slot } i\}$,
$H_n := \{s \text{ sends packet } j \text{ to } r \text{ at the } n^{th} \text{ selection in time-slot } i\}$.

The probability π_i^j can then be expressed as

$$\begin{aligned} \pi_i^j &= 1 - \prod_{n=1}^{c} (1 - P(H_n|E) \cdot P(E)) = \\ &= 1 - \cdot \prod_{n=1}^{c} (1 - P(D|C \cdot E \cdot A) \cdot P(C|E \cdot A) \cdot P(A) \cdot P(E)). \end{aligned} \tag{9}$$

We have that $P(A) = \xi_{i-1}^j$. Let us now calculate π_i^j assuming that r is a contributor. We then have that $P(E) = 1 - p$, so we proceed in finding the value of $P(C|E \cdot A)$. The probability that r will be chosen by s depends on the total number of eligible contributing neighbors of s, since given E, s will only consider contributors as possible recipients at slot i. A contributing neighbor of s is eligible if it does not have packet j, or if it has packet j but it misses at least one other packet that s possesses. So, the probability p_e is written as

$$p_e = 1 - \xi_{i-1}^j + \xi_{i-1}^j \cdot \Big(1 - \prod_{z=\ell(i), z \neq j}^{i-1} \big(1 - \xi_{i-1}^z \cdot (1 - \xi_{i-1}^z)\big)\Big) \tag{10}$$

Let us denote by K the r.v. denoting the number of eligible neighbors of s excluding r. Given that s has D^c contributing neighbors in total, the probability that r will be chosen by s as a recipient of the next packet is expressed as

$$P(C|E \cdot A) = \sum_{k=0}^{D^c-1} \frac{1}{k+1} \cdot P(K = k), \tag{11}$$

where K follows the binomial distribution with parameters $(p_e, D^c - 1)$.
Next, we calculate $P(D|E \cdot C \cdot A)$, the probability that s will send packet j to r, given that s has packet j and it chooses to send a packet to r. This probability is independent of the rank that r was chosen within the slot. We denote by Z the r.v. representing the number of eligible packets of peer s with respect to r. The probability, then, that packet j is sent is

$$P(D|C \cdot E \cdot A) = \sum_{z=1}^{i-\ell(i)} \frac{1}{z} P(Z = z). \tag{12}$$

In the following define a recursion to calculate the distribution of Z. We define the probability vector

$$Q_s = \{\xi_i^{\ell(i)} \cdots \xi_i^{j-1}, 1, \xi_i^{j+1} \cdots \xi_i^{i-1}\},$$

which contains the probabilities that node s has packets at the different buffer positions, given that it has packet j. Similarly, we define the probability vector of peer r, given that it is a contributor.

$$Q_r = \{\xi_i^{\ell(i)} \cdots \xi_i^{j-1}, 0, \xi_i^{j+1} \cdots \xi_i^{i-1}\}$$

We initialize the recursion by setting $L_{0,0} = 1$ and $L_{z,0} = 0$ for $z < 0$. The recursion is then defined by

$$\begin{aligned} L_{z,n} \quad = \quad & L_{z,n-1}(1 - Q_s(n)(1 - Q_r(n))) + \\ & L_{z-1,n-1} Q_s(n)(1 - Q_r(n)). \end{aligned} \tag{13}$$

The recursion is executed for $n = 1 \ldots B - 1$ and for every n for $z = 0 \ldots l$. The distribution of Z is then given as $P(Z = z) = L_{z,i-\ell(i)}$ for for $z = 1 \ldots i - \ell(i)$.
We can calculate π_i^j for the case where r is non-contributor following the same steps as we did for the case where r is a contributor. In that case, however, we have that:

- $P(E) = p$
- $p_e = 1 - \phi_{i-1}^j + \phi_{i-1}^j \cdot \left(1 - \prod_{z=\ell(i), z\neq j}^{i-1} \left(1 - \phi_{i-1}^z \cdot (1 - \phi_{i-1}^z)\right)\right)$
- $Q_r = \{\phi_i^{\ell(i)} \cdots \phi_i^{j-1}, 0, \phi_i^{j+1} \cdots \phi_i^{i-1}\}$

By plugging these values into eq. (9), (10), (13), we derive π_i^j for a non-contributing peer r.

Paper E

Playout Adaptation for Peer-to-Peer Streaming Systems under Churn

Ilias Chatzidrossos and György Dán.
Submitted to Packet Video Workshop (PV) 2012.

Playout Adaptation for Peer-to-Peer Streaming Systems under Churn

Ilias Chatzidrossos and György Dán
School of Electrical Engineering
KTH, Royal Institute of Technology, Stockholm, Sweden
Email: {iliasc, gyuri}@kth.se

Abstract

We address the problem of playout adaptation in peer-to-peer streaming systems. We propose two algorithms for playout adaptation: one coordinated and one distributed. The algorithms dynamically adapt the playback delay of the peers so that the playout miss ratio is maintained within a predefined interval. We validate the algorithms and evaluate their performance through simulations under various churn models. We show that playout adaptation is essential in peer-to-peer systems when the system size changes. At the same time, our results show that distributed adaptation performs well only if the peers in the overlay have similar playback delays. Thus, some form of coordination among the peers is necessary for distributed playout adaptation in peer-to-peer streaming systems.

1 Introduction

Playout adaptation is widely used in multimedia communication to provide continuous playback of audio and/or video information despite network induced impairments, such as delay jitter and fast varying network throughput. To make playout adaptation possible, the receiver stores the received data in a playout buffer before eventually playing it out. Playout adaptation consists then of two tasks. At the beginning of a connection the receiver has to decide when to start the playout and potentially what to start the playout with; later on it aims to avoid that the playout buffer becomes empty or that it overflows by, for example, adapting the playback rate [1] or by rescheduling the playback of the subsequent talksuprts during silence periods [2].

In peer-to-peer (P2P) streaming systems, the audio or video data sent by a server can be forwarded by several peers before it reaches a particular peer. While data forwarding saves server upload bandwidth and cost, it makes the data delivery process subject to several factors that are hard to control or to predict. On one

hand, the forwarding algorithms used by the peers together with the varying upload bandwidths of the peers result in a complex multi-path data distribution process, in which out-of-order data delivery is not the exception but the rule [3]. On the other hand, the peer arrivals and departures disturb the data forwarding and also influence the time needed to distribute the data through changing the number of peers in the system [4].

Thus, playout adaptation in P2P systems should not lead to unnecessary adaptation, despite the highly stochastic nature of the data distribution process. At the same time, it must respond promptly to changes in the chunk missing ratio of the peers, caused by, for example a sudden increase in the number of peers. To meet these challenges, playout adaptation in P2P streaming systems can potentially make use of more information than in the case of point-to-point communication. To support playout adaptation, the peers in the system might use information from their neighbors, or could rely on global information, obtained from a central or from a distributed entity. Nevertheless, the playout adaptation performed by individual peers might affect the data delivery to other peers, thus the problem is inherently complex.

In this paper we address the problem of playout adaptation in P2P live streaming systems. We propose a centralized and a distributed algorithm to adapt the playback delay such as to maintain the system in a desired operating regime. We use extensive simulations to show that the algorithms are beneficial in a steady state system, and they provide significant gains when the number of peers changes.

The rest of the paper is organized as follows. In Section 2 we review related work. In Section 3 we describe the considered P2P streaming system and formulate the playout scheduling and adaptation problem and in Section 4 we present the two adaptation algorithms. Section 5 provides a simulation-based evaluation of the algorithms. Section 6 concludes the paper.

2 Related work

Playout scheduling and adaptation has received significant attention for the case of point-to-point media transmission. For streaming media, playout schedulers rely on time or on buffer occupancy information [5]. In time-based playout schedulers, timing information is embedded in the packets so that the receiving host can measure absolute or relative differences in the packet delivery times [6]. In buffer-based playout schedulers the network's condition is inferred from the evolution of the buffer occupancy [7].

Playout adaptation can be performed in various ways, depending on the type of media. In the case of voice communications, schedulers can leverage the silence periods and postpone the playout of the next talkspurt to ensure playout continuity [2]. In the case of video/audio streaming, playout adaptation can be performed by either frame (packet) dropping or by altering the frames' playout duration. In

the case of frame dropping, the receiver drops frames from the playout buffer to avoid buffer overflow or freezes the playout to avoid buffer underflow [8, 9]. By altering the frame duration the receiver can decrease the rate of the playout so that the buffer builds up and can decrease the playback delay if the playout buffer occupancy is high [1, 7]. In [1], a window-based playout smoothing algorithm changes the playout rate at the receiver based on the current buffer occupancy and on the traffic during the next time window as predicted by a neural network. In terms of objectives our work is more similar to [7], where an adaptive media playout algorithm selects the playout rate according to the channel conditions in order to achieve low latency for a given buffer underflow probability.

Contrary to point-to-point communication, playout adaptation for P2P streaming has not received much attention. In [9], the authors compare delay-preserving and data-preserving playout schedulers and conclude that a playout scheduler should keep the peers synchronized to improve the data exchange efficiency. They consider however, tree-based systems where there are only peer arrivals. In [10], a playout adaptation scheme for tree-based systems is proposed that allows peers to adjust their playout rate in order to reduce the average playback delay in the system. Two processes performed locally at the peers, catching-up and parent-reversal, are defined so that the propagation delay in the resulting overlay trees is minimized. Nevertheless, none of these works consider the need for playout adaptation due to the changes of the overlay size. To the best of our knowledge, our work is the first that addresses the playout scheduling and adaptation problem for P2P streaming and focuses on mesh-based streaming systems, which constitute the vast majority of deployed P2P streaming systems nowadays.

3 Problem Formulation

We consider a P2P live streaming system consisting of a streaming server and a set $\mathcal{N}(t)$ of peers. Peers arrive to the system according to a counting process $A(t)$, and depart with intensity $\mu(t)$. We denote the number of peers at time t by $N(t)$, the arrival time of peer n by T_A^n and its departure time by T_D^n. The streaming server and the peers maintain an overlay, and use the overlay connections to distribute the video stream generated by the streaming server to the peers. The video stream consists of a sequence of chunks of data, e.g., data corresponding to a few frames or a group of frames, generated at regular time intervals δ, and we denote the time when chunk c is generated at the streaming server by t_c. To distribute the stream, the peers relay the chunks among each other according to some forwarding algorithm, e.g., [11, 12]. In order to make relaying possible, each peer stores the chunks that it has received in a buffer until the chunks are played out.

We denote the time when chunk c is received by peer n by t_c^n. The time $t_c^n - t_c$ it takes for chunk c to be delivered to peer n depends on many factors, such as the number of peers, the overlay structure, the peers' upload rates, the fowarding

algorithm, and is in general hard to predict. Consider now an arbitrary peer n and the sequence of chunk arrival times $t_c^n, t_{c+1}^n \ldots$. Peer n can only play out chunk c after it receives it, i.e., at some time $t \geq t_c^n$. We refer to the difference $t - t_c = B^n(t_c^n)$ as the playback delay used for chunk c. We refer to the share of chunks that a peer cannot play out on time as the chunk missing ratio. For peer n and a time interval $\iota = [T_1, T_2]$ we define the chunk missing ratio as

$$\pi^n(\iota) = \frac{|\{c : t_c^n \in \iota \cap [T_A^n, T_D^n] \wedge B^n(t_c^n) < t_c^n - t_c\}|}{|\{c : t_c^n \in \iota \cap [T_A^n, T_D^n]\}|} \tag{1}$$

Furthermore, we define the average chunk missing ratio $\pi(\iota)$ calculated over all peers in the system in the interval ι.

The playback delay $B^n(t)$ has to be chosen upon arrival, but need not be constant until the departure of the peer. As shown by recent work [13], the perceived visual quality is rather insensitive to minor variations of the playout rate. According to the model of perceived visual quality proposed in [13], the mean opinion score (MOS) can be expressed as a function of the frame rate f as

$$MOS(f) = Q_{max} \frac{1 - e^{-\gamma \frac{f}{f_m}}}{1 - e^{-\gamma}}, \tag{2}$$

where γ and Q_{max} are video specific constants and f_m is the maximum frame rate. Consider now that the frame rate is changed to $f_a = (1 + a)f$ for some small $a \approx 0$. For $a \approx 0$ the derivative of the MOS score can be approximated by

$$\frac{dMOS(f_a)}{da} \approx Q_{max} \gamma \frac{f}{f_m} \frac{e^{-\gamma \frac{f}{f_m}}}{1 - e^{-\gamma}}, \tag{3}$$

by using that $e^a|_{a=0} = 1$. Observe that if (2) is high then (3) is low, i.e., the perceived quality is not very sensitive to small variations of the playback rate. We thus consider that the playback delay $B^n(t)$ of a peer can be adapted by accelerating or decelerating the playback. We denote the maximum rate of adaptation by $\hat{a} > 0$, and assume that the effect of adaptation on the perceived visual quality is negligible at adaptation rates $|a| < \hat{a}$, similar to the assumption in [1, 7].

For peer n to be able to play out all chunks, it has to adapt its playback delay $B^n(t_c^n)$ such that $t_c + B^n(t_c^n) \geq t_c^n$ for all chunks c that it should play out in the time interval $[T_A^n, T_D^n]$. If all peers adapt their playback delay this way then $\pi = 0$, but this might be difficult to achieve in practice due to the dynamics of the system. Thus, we consider that a chunk missing ratio no higher than $\tau > 0$ is acceptable, and can be compensated for by some form of channel coding. At the same time if channel coding capable of compensating for a chunk missing ratio of τ is used, then the actual chunk missing ratio should not be below $\eta\tau$, where $1 > \eta > 0$, because then the bandwidth used for channel coding would be wasted. Thus, we consider that the goal of playout adaptation is to adjust the playback delay $B^n(t)$ of the

peers such that the chunk missing ratio stays within the band $[\eta\tau, \tau]$, subject to the constraint on the maximum rate of adaptation $\hat{a}$.

We measure the performance of playout adaptation over a time interval $\iota = [T_1, T_2]$ by the square error of the chunk missing ratio calculated over the interval ι,

$$SE(\iota) = ([\eta\tau - \pi(\iota)]^+)^2 + ([\pi(\iota) - \tau]^+)^2 \quad (4)$$

where $[.]^+$ denotes the positive part. When the SE is averaged over consecutive intervals ι, then we get the mean square error (MSE) of the chunk missing ratio. The MSE quantifies the average deviation of the chunk missing ratio from the target interval.

4 Playout Adaptation Algorithms

In the following we present two playout adaptation algorithms. The first one, called *Coordinated Adaptation (CA)*, is a centralized algorithm and lets all peers have the same playback delay calculated based on the average chunk missing ratio. The second one, called *Distributed Adaptation (DA)*, allows each peer to adapt its playback delay based on the chunk missing ratio it experiences.

4.1 Coordinated Adaptation (CA)

In the case of coordinated adaptation the playback delay is recalculated periodically by executing the adaptive band control algorithm shown in Algorithm 1, and is the same for all peers in the system.

At the i^{th} execution of the algorithm, at time T_i, the input to the control algorithm is the average chunk missing ratio $\pi(\iota)$ over the interval $\iota = [T_i - B(T_i), T_i]$. This is used to calculate the exponentially weighted moving average $\overline{\pi}$ of the chunk missing ratio and its deviation $\overline{\sigma}$ (lines 2-3 in Algorithm 1), similar to the round-trip time estimation in TCP [14], using $\alpha = 0.125$ and $\beta = 0.25$. The algorithm uses $\overline{\pi}$, $\overline{\sigma}$, and the most recent average chunk missing ratio $\pi(\iota)$, to decide how to change the playback delay. The playback delay B is increased if both the average chunk missing ratio and its moving average exceed the target τ or if the average chunk missing ratio exceeds the target τ and is higher than the moving average plus a constant $\kappa > 0$ times its deviation (lines 6-12). The second condition resembles the RTO calculation used in TCP [14]. By tuning κ we can control the trade-off between fast adaptation to changing conditions and unnecessary adaptation in a stationary system. The conditions for decreasing the playback delay are similar (lines 15-18).

Adaptation is performed in time units equal to the chunk time δ. The amount of time units γ by which the playback delay is increased or decreased is determined by the history of adaptations performed by the algorithm. Initially, $\gamma = 1$. This value is doubled every time the playback delay has to be increased consecutively, and is decremented by one otherwise. The adaptation of γ resembles the additive increase

Algorithm 1 Adaptive band control algorithm for CA

```
1: Input: π(ι)  for  ι = [T_i − B(T_i), T_i]
2: σ̄ = (1 − β) · σ̄ + β · |π̄ − π(ι)|,  (β = 1/4)
3: π̄ = (1 − α) · π̄ + α · π(ι),  (α = 1/8)
4: if π(ι) > τ then
5:    if π̄ > τ OR π(ι) > π̄ + κ · σ̄ then
6:       if previous_action == INCREASE then
7:          γ = 2 · γ
8:       else
9:          γ = max(γ − 1, 1)
10:      end if
11:      B′ = B + γ · δ
12:      previous_action ← INCREASE
13:   end if
14: else if π(ι) < η · τ then
15:   if π̄ < η · τ OR π(ι) < π̄ − κ · σ̄ then
16:      γ = max(γ − 1, 1)
17:      B′ = B − δ
18:      previous_action ← DECREASE
19:   end if
20: else
21:   γ = max(γ − 1, 1)
22: end if
```

multiplicative decrease used in TCP congestion control [14], and allows for a fast increase of the playback delay if needed, but leads to a smaller rate of decrease.

If the playback delay is changed, the peers are informed about the new playback delay B', and accelerate or decelerate their playout rate to adapt their playback delay to B'. The time needed to perform the adaptation is $|B' - B|/\hat{a}$. The next time the control algorithm is executed is time B' after the adaptation has finished, that is, at time $T_{i+1} = T_i + |B' - B|/\hat{a} + B'$. The arriving peers set their playback delay to $B^n(T_A^n) = B'$ upon arrival and then adapt their playback delay as dictated by the *CA* algorithm.

4.2 Distributed Adaptation (DA)

In the case of distributed adaptation the playback delay is recalculated periodically by every peer through executing the band control algorithm shown in Algorithm 2. The input to the control algorithm is the length s_j of the last eight loss intervals (i.e., the number of chunks received between missed chunks), which is used to estimate the average chunk missing ratio using the Average Loss Intervals method as in TFRC [15].

Algorithm 2 Band control algorithm for DA

$w \leftarrow (1, 1, 1, 1, 0.8, 0.6, 0.4, 0.2)$
$\hat{s} \leftarrow \frac{\sum_{j=1}^{8} w_j \cdot s_j}{\sum_{j=1}^{8} w_j}$
$\pi = \frac{1}{\hat{s}+1}$
if $\pi > \tau$ **then**
  if $previous\_action == INCREASE$ **then**
    $\gamma = 2 \cdot \gamma$
  else
    $\gamma = max(\gamma - 1, 1)$
  end if
  $B' = B + \gamma \cdot \delta$
  $previous\_action \leftarrow INCREASE$
else if $\pi < \eta \cdot \tau$ **then**
  $\gamma = max(\gamma - 1, 1)$
  $B' = B - \delta$
  $previous\_action \leftarrow DECREASE$
else
  $\gamma = max(\gamma - 1, 1)$
end if

The algorithm uses the estimated chunk missing ratio to decide how to change the playback delay. The playback delay is increased if the average chunk missing ratio exceeds the target τ, is decreased if the average chunk missing ratio is lower than $\eta\tau$, and is left unchanged otherwise. The adaptation step size γ is adjusted the same way as in the case of Algorithm 1, i.e., it is doubled upon consecutive increases of the playback delay, and is decremented by one unit otherwise. Once the new playback delay B' is calculated, the peer accelerates or decelerates its playout rate to adapt its playback delay to the new value. The time needed for adaptation is $|B' - B|/\hat{a}$. The next time the control algorithm is executed is time B' after the adaptation has finished, similar to *CA*.

We consider three policies that the peers can use to choose their initial playback delay $B^n(T_A^n)$ upon arrival to the system.

- Global: The initial playback delay is the average playback delay of the peers in the overlay, i.e., $B^n(T_A^n) = \frac{1}{|\mathcal{N}(T_A^n)|} \sum_{j \in \mathcal{N}(T_A^n)} B^j(T_A^n)$
- Local: The initial playback delay is the average playback delay of the neighboring peers, i.e., $B^n(T_A^n) = \frac{1}{|\mathcal{D}_n(T_A^n)|} \sum_{j \in \mathcal{D}_n(T_A^n)} B^j(T_A^n)$, where $\mathcal{D}_n(T_A^n)$ is the set of neighbors of peer n.
- Fixed: The initial playback delay is $B^n(T_A^n) = B_0 > 0$.

Class	Upload (Mbps)	Peers (%)
1	5	10
2	1.5	10
3	1	40
4	0.55	40

Table 1: Peer Bandwidth Classes

Of these three policies, the Local policy is easiest to implement in a distributed way: An arriving peer can schedule for playback the chunk that its average neighbor would play out. The other two policies would require a globally synchronized clock, and therefore we use them mainly as a basis for comparison.

5 Performance evaluation

In the following we use simulations to evaluate the proposed algorithms and to get an insight into their operation.

5.1 Simulation Methodology

We implemented the playback adaptation algorithms in the publicly available packet-level event-driven simulator P2PTV-SIM [16]. We implemented an overlay management algorithm that maintains an overlay graph in which every peer has between $0.5d$ and d neighbors: arriving peers try to establish d connections to randomly selected peers, potentially by letting some peers drop one of their already established connections. If the number of neighbors of a peer drops below $0.5d$ then it tries to connect to new neighbors. In our simulations, we use $d = 20$.

The video stream has a rate of 1 Mbps and is segmented into equally sized chunks of 100 kbits, having thus $\delta = 0.1$ s. Chunk forwarding between the peers is based on the pull-token algorithm from [12]. The download bandwidth of the peers is unlimited, while for the upload bandwidth, we use the distribution shown in Table 1. In order to capture the effect of network delays, peers are dispersed over seven geographical regions according to the network model presented in [17] with an average latency between a pair of peers of 96 ms. For the adaptation algorithms *CA* and *DA*, we use $\hat{a} = 0.05$ and a value of $\kappa = 2$, which we found to yield a good trade-off between fast response to the chunk missing ratio and unnecessary adaptation of the playback delay.

We consider two models of peer churn. In the *Markovian churn* model the peer arrival process is a Poisson process with arrival intensity $\lambda(t)$, and the peer holding time distribution is exponential with mean $1/\mu$. In the *Flash-crowd churn* model N_f peers arrive according to a homogeneous Poisson process with intensity λ_f over

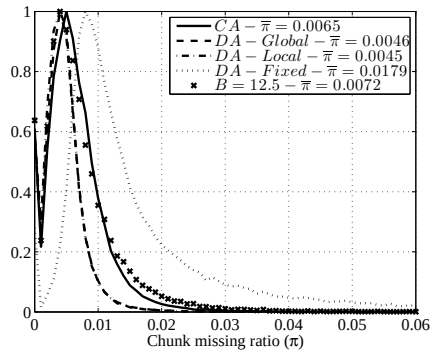


Figure 1: Normalized histogram of the chunk missing ratio for the *CA* algorithm and the *DA* algorithm with different startup policies. The target miss ratio is $\tau = 0.01$ and $\eta = 0.5$. Overlay of $\overline{N} = 500$ peers.

a time interval $[t_f, t_f + N_f/\lambda_f]$, and the peers remain in the overlay until the end of the simulation.

5.2 Playout Adaptation in Steady State

The first scenario we consider is a system in steady state, generated using the Markovian churn model. The peer arrival rate is $\lambda(t) = 1.66/s$ and the mean holding time is $1/\mu = 300$ s. The average number of peers is thus $\overline{N} = 500$.

Fig. 1 shows the normalized histogram of the chunk missing ratio of the peers after a warm-up period of 2000 seconds in a simulation of 6000 seconds, for the *CA*, *DA* algorithms and without playout adaptation (*NA*). For *NA*, we used a fixed playback delay of $B = 12.5$ s, which is the average playback delay obtained using the *CA* algorithm and $\tau = 0.01$, $\eta = 0.5$. Fig. 2 shows the cumulative distribution function of the playback delay for the same time interval.

The results show that even in the case of a system in steady state, adaptation decreases the occurence of high chunk missing ratios (the tail of the histogram compared to the system without adaptation). Comparing the results for the *CA* and for the *DA* algorithm with different policies we observe that the *Fixed* policy not only leads to high chunk missing ratios (Fig. 1) but it also leads to high playback delays (Fig. 2). The *Global* and the *Local* policies, however, lead to lower chunk missing ratios than the *CA* algorithm at the price of higher playback delays. The reason is that using *DA* it is not the *average* chunk missing ratio that is maintained in the target band, but it is each *invididual* peer's chunk missing ratio.

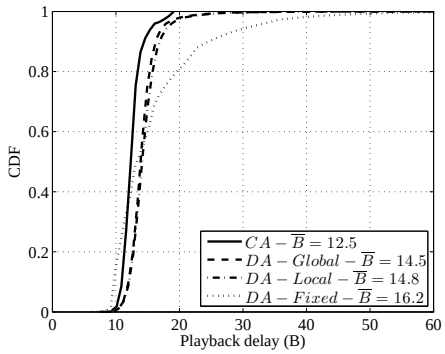


Figure 2: CDF of the playback delays for the *CA* and the *DA* algorithms with different startup policies. The average playback delays are shown in the legend. The target miss ratio is $\tau = 0.01$ and $\eta = 0.5$. Overlay of $\overline{N} = 500$ peers.

5.3 Adaptation in a Non-stationary System

The second scenario we consider is a non-stationary system, generated using the Markovian churn model in the following way. The total simulation time is 12000 s, and the peer arrival intensity is $\lambda(t) = 1.66/s$ for the time intervals $[0, 3000)$ and $[7000, 12000]$, and it is $\lambda(t) = 16.66/s$ for the time interval $[3000, 7000)$. The average peer holding time is $1/\mu = 300$ s. Thus, there is a transition from an overlay of $\overline{N} = 500$ peers to an overlay of $\overline{N} = 5000$ peers, and then back to an overlay of $\overline{N} = 500$ peers.

Fig. 3 shows the square error (SE) calculated over consecutive intervals of 50 seconds for the *CA* and *DA* algorithms and without adaptation (*NA*). The figure also shows the mean square error ($\overline{MSE}$) for the entire simulation. The SE for the *CA* algorithm is very small, which means that the algorithm manages to keep the chunk missing ratio within the interval $[\eta\tau, \tau]$ with small deviations, despite the increase in the overlay size by a factor of ten. Without adaptation the chunk missing ratio increases significantly during the time period with high arrival intensity. We also notice an increase in the SE for the *DA* and *NA* systems right after the the switch from the high arrival intensity to the low arrival intensity period ($t = 7000$ s). For the system without adaptation the degradation is expected, since it takes some time for the peers to acquire new neighbors and maintain an adequate download rate.

For the system employing the *DA* algorithm, the explanation for the performance degradation lies in the algorithm itself. If peers do not change their playback delay in a coordinated manner, but based on local information, clusters of peers with similar playback delays emerge. When the departure rate in the overlay is higher than the arrival rate, peers that are losing their neighbors try to find new ones but the new neighbors may have different playback delays, i.e., only partially overlapping

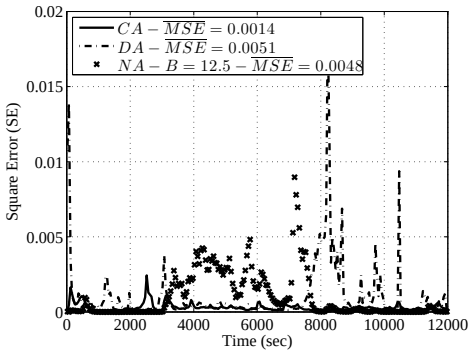


Figure 3: SE of the chunk missing ratio. The target chunk miss ratio is $\tau = 0.01$ and $\eta = 0.5$ which means that the SE is 0 when $\pi(t) \in [0.005, 0.01]$. The MSE is also shown in the legend.

buffers, decreasing thus the efficiency of the data exchange. This is the reason why the same degradation is not observed when using the *CA* algorithm that keeps peers synchronized. It is therefore necessary to use some form of coordination among peers when performing playout adaptation in a distributed manner in order to maintain similar playback delays for all the peers in the overlay.

Next, we show the gain of playout adaptation in terms of the video distortion. We quantify the video distortion through the Peak Signal-to-Noise Ratio (PSNR) of the received video. The PSNR is defined as $PSNR = 10 \cdot log_{10}(255^2/D)$, where D is the total (source and channel) distortion. In order to calculate the distortion of the video as a function of the chunk missing ratio, we use the loss-distortion model presented in [18].

Fig. 4 shows the average PSNR over time, calculated for the Foreman sequence for the non-stationary system. During the initial low arrival intensity period the performance with and without adaptation is similar. When the overlay expands, the PSNR degrades significantly if no adaptation is used, the difference varying between 2 and 4 dB, compared to the overlays using adaptation. The *CA* algorithm manages to maintain the PSNR almost constant for the whole simulation time and the transitions from the small to large overlay and back are indistinguishable. The *DA* algorithm performs well too but, as also shown earlier, its PSNR degrades sharply (up to 6 dB) after the transition from the high arrival intensity to the low arrival intensity period starts. Furthermore, the PSNR of the system employing the *DA* algorithm exhibits large variations after $t = 7000$ s, which can have a significant impact on the perceived video quality.

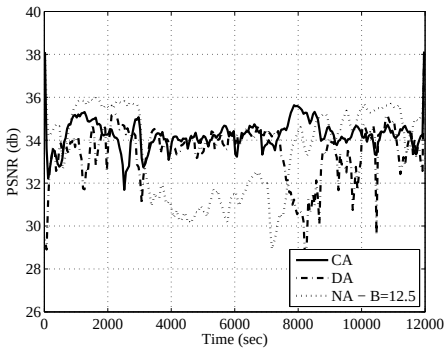


Figure 4: PSNR of the received video over time for the *CA*, *DA* algorithms and for the an overlay without playout adaptation.

5.4 Adaptation under Flash Crowd Scenarios

Finally, we evaluate the adaptation algorithms under a flash crowd scenario. We study the transition from an overlay of $N_1 = 500$ peers to an overlay of $N_2 = 5000$ peers at different peer arrival intensities. We simulate the flash-crowd scenario by superposing two processes. The first process is generated using the *Markovian churn* model, with arrival rate $\lambda(t) = 1.66/s$ for the whole duration of the simulation. The peer holding times are exponentially distributed with mean $1/\mu = 300$ seconds. The second process is generated using the *Flash-crowd churn* model with $t_f = 3000$ s and arrival rates $\lambda_f = 20/s$, $\lambda_f = 40/s$ and $\lambda_f = 60/s$ to generate a small, moderate and severe flash crowd, respectively. The total simulation time is 6000 seconds.

Fig. 5 shows the square error (SE) of the chunk missing ratio calculated over consecutive intervals of 50 seconds. For the period before the flash crowd the SE lies in the same region as in Fig. 3. During the flash crowd though, there is a sharp increase in the SE, whose peak is higher with higher λ_f. The important thing to observe here is that the *CA* algorithm manages to converge to the desired band within a short period after the flash crowd. The time it takes to return the chunk missing ratio to the desired band does not seem to depend on λ_f, which indicates that the *CA* algorithm adapts well to different levels of impairment by appropriately increasing the adaptation step size.

6 Conclusion

In this work we addressed the problem of playout adaptation in P2P streaming systems. We defined two algorithms that perform playout adaptation. Using co-

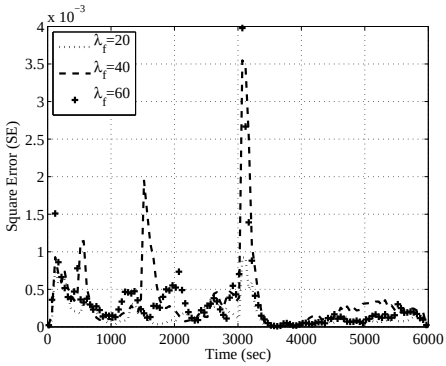


Figure 5: SE of the chunk missing ratio for different arrival intensities during the flash crowd. The target chunk miss ratio is $\tau = 0.01$ and $\eta = 0.5$.

ordinated adaptation peers adapt their playback delay synchronously, maintaining fully overlapping playout buffers. Using distributed adaptation each peer adapts its playback delay on its own, without any coordination with neighbors. We used extensive simulations to validate the algorithms and to evaluate their performance under various churn models. We showed that playout adaptation does not decrease the system's performance in steady state and that it is essential when the overlay size is changing. Our results indicate that peers can perform playout adaptation based solely on information about the playout position of their neighbors, which can be easily obtained locally. Nevertheless, our results show that distributed playout adaptation works well only if peers maintain similar playback delays across the whole overlay, thus some form of coordination is necessary.

References

[1] M.C. Yuang, Po L. Tien, and Shih T. Liang. Intelligent video smoother for multimedia communications. *IEEE Journal on Selected Areas in Communications*, 15(2):136 –146, Feb. 1997.

[2] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *IEEE INFOCOM*, pages 680 –688, jun 1994.

[3] I. Chatzidrossos, Gy. Dán, and V. Fodor. Delay and playout probability tradeoff in mesh-based peer-to-peer streaming with delayed buffer map updates. *P2P Networking and Applications*, 3:208–221, March 2010.

[4] Gy. Dán and V. Fodor. Delay asymptotics and scalability for peer-to-peer live streaming. *IEEE Transactions on Parallel and Distributed Systems*, 20(10):1499–1511, October 2009.

[5] N. Laoutaris and I. Stavrakakis. Intrastream synchronization for continuous media streams: a survey of playout schedulers. *Network, IEEE*, 16(3):30 –40, may/jun 2002.

[6] Werner Geyer, Christoph Bernhardt, and Ernst Biersack. A synchronization scheme for stored multimedia streams. In *Interactive Distributed Multimedia Systems and Services IDMS'96*, pages 277–295. Springer Verlag, 1996.

[7] M. Kalman, E. Steinbach, and B. Girod. Adaptive media playout for low-delay video streaming over error-prone channels. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(6):841–851, june 2004.

[8] E. Biersack, W. Geyer, and C. Bernhardt. Intra- and inter-stream synchronisation for stored multimedia streams. In *Proc. of the Third IEEE International Conference on Multimedia Computing and Systems*, pages 372 –381, jun 1996.

[9] C. Vassilakis, N. Laoutaris, and I. Stavrakakis. On the impact of playout scheduling on the performance of peer-to-peer live streaming. *Computer Networks*, 53:456–469, March 2009.

[10] H. Jiang and S. Jin. NSYNC: Network synchronization for peer-to-peer streaming overlay construction. In *Proc. of NOSSDAV*, pages 1–6, 2006.

[11] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: Optimal performance trade-offs. In *Proc. of ACM SIGMETRICS*, 2008.

[12] A. Carta, M. Mellia, M. Meo, and S. Traverso. Efficient uplink bandwidth utilization in P2P-TV streaming systems. In *Proc of GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, pages 1–6, December 2010.

[13] Y.F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang. Modeling the impact of frame rate on perceptual quality of video. In *Proc. of 15th IEEE International Conference on Image Processing (ICIP)*, pages 689–692, 2008.

[14] Computing TCP's retransmission timer - IETF RFC 2988, November 2000.

[15] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM*, pages 43–56, August 2000.

[16] P2PTV-SIM, http://napa-wine.eu/cgi-bin/twiki/view/public/p2ptvsim.

[17] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs. In *Proc. of the 10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1 –10, August 2010.

[18] V. Vukadinovic and Gy. Dán. Multicast scheduling for scalable video streaming in wireless networks. In *ACM Multimedia Systems (MMSys)*, February 2010.

Paper F

Small-world Streaming: Network-aware Streaming Overlay Construction Policies for a Flat Internet

Ilias Chatzidrossos, György Dán and Arnaud Legout.
Submitted to International Conference on Distributed Computing Systems (ICDCS) 2012.

Small-world Streaming: Network-aware Streaming Overlay Construction Policies for a Flat Internet

Ilias Chatzidrossos[1], György Dán[1] and Arnaud Legout[2]

[1]KTH, Royal Institute of Technology, Stockholm, Sweden
[2]INRIA Sophia-Antipolis, France

Abstract

Recent measurements indicate that the peering agreements between Autonomous Systems (AS) are flattening the AS level topology of the Internet. The transition to a more flat AS topology opens up for new possibilities for proximity-aware peer-to-peer overlay construction. In this paper we consider the problem of the construction of overlays for live peer-to-peer streaming that leverage peering connections to the maximum extent possible, and investigate how a limited number of overlay connections over transit links should be chosen such as to maximize the streaming performance. We define a set of transit overlay link establishment policies that leverage topological characteristics of the AS graph. We evaluate their performance over regular AS topologies using extensive simulations, and show that the performance difference between the policies can be up to an order of magnitude. Thus, it is possible to maximize the system performance by leveraging the characteristics of the AS graph. Based on our results we also argue that the average loss probability is not an adequate measure of the performance of proximity-aware overlays. We confirm our findings via simulations over a graph of the peering AS topology of over 600 ASs obtained from a large measurement data set.

1 Introduction

Peers establish connections between each other in peer-to-peer (P2P) systems to form an overlay over the physical network. Depending on the geographic distribution of the peers, the overlay can span a large number of Internet Service Providers (ISP). The connections among the peers together with the data scheduling algorithms determine the flow of data through the overlay, from some sources to the peers, and hence between the ISPs. Thus, overlays that are constructed unaware of the underlying network's topology can cause unnecessary inter-ISP traffic [1].

A natural way to decrease the amount of inter-ISP traffic generated by P2P systems is to establish connections by preference between peers within the same ISP, that is, *locality-aware* overlay construction. In order to facilitate locality-aware overlay construction, ISP-managed services, e.g., ALTO [2], are being developed to provide network topology and cost information. Nevertheless, several studies showed that locality-awareness leads to overlays in which peers form clusters, and a mixture of locality-aware and random connection establishment is needed to achieve a reasonable trade-off between performance and the amount of traffic delivered over inter-ISP links, both for P2P file-sharing [3, 4, 5, 6] and for P2P live streaming [7, 8, 9].

The trade-off between locality-awareness and good P2P system performance is analogous to the trade-off between the clustering coefficient and the characteristic path length of sparse graphs [10, 11]. Recent results in algorithmic graph theory show that starting from a highly clustered graph with high characteristic path length, it is possible to obtain a clustered graph with low characteristic path length by rewiring a few edges only [11]. Since an overlay with a low characteristic path length would allow fast chunk distribution, these results provide theoretical support for the trade-offs observed in locality-aware P2P systems [4, 5, 6, 8, 9]. They do, however, also raise two fundamental questions that have not been addressed in the context of P2P systems. First, given a network topology, how could one engineer a matching overlay topology that allows for good P2P system performance. Second, given a network topology and a fixed amount of traffic cost to be generated, how should one engineer an overlay topology to maximize the system's performance.

We address these two questions by relying on recent studies of the Internet's autonomous system (AS) level topology [12, 13] and on results from algorithmic graph theory [11]. Recent studies show that *peering links* between ISPs have become more prevalent over the last few years and are flattening the AS-level topology of the Internet [12, 13]. The importance of peering links is that they are established between ISPs for mutual benefit, and the traffic exchanged over them is *not paid* for. This is in contrast with *transit links* between a customer and a provider ISP, where the customer ISP pays for the traffic exchanged with its provider ISP based on, e.g., the 95% rule.

Given that topology information is available (e.g., from ALTO [2]), in this work we address the problem of strategically forming network-aware P2P overlays. Our contributions are threefold. First, inspired by results in algorithmic graph theory we define policies for constructing overlay topologies that adapt to the underlying network topology and provide good system performance. Second, we show that the performance difference between the resulting overlays can be more than an order of magnitude for the same amount of transit inter-ISP traffic. Furthermore, policies that establish transit connections between peers in distant ISPs perform better in general. Third, we show that the optimal policy does not only depend on the network topology but also on the location of the streaming server, and we show evidence for that higher order statistics of the loss probabilities might be

needed for a fair comparison of proximity-aware streaming systems. These results give important insight for future network-aware overlay design and performance evaluation.

The rest of the paper is organized as follows. In Section 2 we review the related work. In Section 3 we describe our model of the AS-level network topology and the P2P overlay, and introduce results from algorithmic graph theory that motivate our work. Section 4 provides the problem formulation and describes the evaluation methodology. In Sections 5 and 6 we show results for regular graph topologies. In Section 7 we show results obtained on a measured Internet AS-level topology. Section 8 concludes the paper.

2 Related work

Locality-awareness in P2P system design was first considered for offline content distribution. A number of works investigated the impact of locality-awareness on the amount of inter-ISP traffic and on the system performance [1, 4, 5, 3, 6], and pointed out that locality-aware neighbor selection leads to clustering and can negatively affect the data distribution performance. To facilitate locality-aware neighbor selection, others focused on the information exchange between ISPs and P2P systems, e.g., P4P [14] and the Oracle service proposed in [5], and are under standardization in the IETF [2].

Locality-awareness for P2P streaming systems has also received some attention recently. A recent measurement study by Ciullo et al. [15] examined four of the most popular P2P streaming systems, and found that locality criteria are only subtly taken into account for overlay creation. Nevertheless, a number of works have addressed the problem of locality-aware P2P streaming. One direction of work has looked at relaying streaming content to relieve the traffic on inter-ISP transit links [16, 17] in a way that considers the peering agreements between ISPs. Nevertheless, the problem of overlay construction was not considered in these works.

The trade-off between streaming performance and peer clustering was considered by exploring different levels of locality-awareness in [7, 8, 9]. Picconi et al. [7] considered the problem of maintaining a target streaming quality in a locality-aware P2P streaming system. In their scheme the peers maintain a set of local and a set of remote connections. They exchange data by preference over local connections, and unchoke remote connections uniformly at random when needed. Jin et al. [8] investigated the streaming performance under connection establishment policies that choose at random between nearby peers and peers with high upload capacity. Seedorf et al. [9] studied the cost savings achievable through the use of the ALTO service through simulations on a real AS topology. Similar to results obtained for BitTorrent [4, 6], these studies found that a fraction of the connections needs to be established to remote peers to avoid performance degradation. Nevertheless, relying on cost and topology information does not necessarily lead to traffic cost

savings. Seedorf et al. [9] found that if the ALTO server of every ISP recommends connections only based on their own costs then even the traffic cost savings become insignificant.

Our work looks at locality-awareness from a novel perspective. First, we investigate how the streaming performance can be maximized for a fixed level of locality-awareness and inter-ISP traffic. This is different from previous works, which explored the trade-off between the amount of inter-ISP traffic and the streaming performance. Second, we account for the recent trend of a flattening AS-level Internet topology [13, 18], and use results from algorithmic graph theory to develop overlay construction policies that are compatible with basic Internet economics, have favorable locality properties but at the same time maximize the streaming performance. Our approach of engineering overlay topologies that match the underlying network topology and maximize streaming performance paves the way for new network-aware P2P system designs.

3 System Model and Motivation

In the following we describe our model of a peer-to-peer streaming overlay spread over a set of ISPs with peering agreements. We then discuss results from algorithmic graph theory that inspired our work.

3.1 System Model

We model the ISP-level *peering* topology with a graph $G = (\mathcal{I}, E)$, where $\mathcal{I}$ is the set of ISPs, and there is an edge $(i, j) \in E$ between vertices i and j if there is a peering agreement between ISPs i and j. We assume that G is connected, that is, there is a path in G between any two vertices i and j. We define the distance of two vertices i and j as the length of the shortest path between i and j in G and denote it by $d(i, j)$. The measured Internet AS-level topology data presented in Section 7 confirm that large connected components of peering ASs already exist in the Internet. At this point let us note that our model is not restricted to ISPs, but can be used for any network region as long as the overlay construction entity is aware of the network regions and can map the peers to those regions. This information could be provided by a dedicated infrastructure like the P4P infrastructure [14]. In particular, when we consider measured topologies in Section 7, we will use the notion of ASs instead of ISPs.

The overlay consists of a set $\mathcal{P}$ of peers with cardinality P. The P peers and the streaming server are spread over the ISPs. We denote by $\iota(p_i)$ the ISP where peer p_i is located. We distinguish between three kinds of overlay connections between the peers. We call a connection between peers $p_i, p_j \in \mathcal{P}$ local if $\iota(p_i) = \iota(p_j)$, we call it peering if $(\iota(p_i), \iota(p_j)) \in E$ and we call it transit otherwise. Figure 1 illustrates an AS-level network topology and the three types of overlay connections.

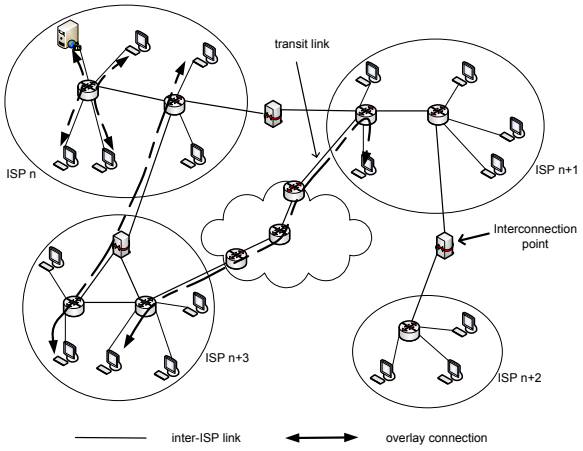


Figure 1: Example of an ISP topology with four ISPs. The thick bidirectional arrows between peers are examples of the three types of overlay connections in P2P overlays: local, peering and transit.

The data chunks can be delivered between the peers through a combination of local, peering and transit connections, and the cost incurred to the ISPs is a function of the amount of data that the peers exchange through these three kinds of overlay connections. In accordance with current charging schemes, we consider that the marginal cost for an ISP is lowest for data that peers exchange over local connections, and it is highest for data that peers exchange over transit connections.

3.2 Proximity-awareness and Small-world Graphs

Intuitively, a proximity-aware P2P overlay should satisfy two criteria in order to provide good streaming quality. First, the overlay path lengths between the peers should be short to allow for fast chunk dissemination. Second, peers in the same ISP should form well-connected clusters to be able to exchange chunks locally. These two requirements correspond to two well-studied characteristics of graphs: the characteristic path length and the clustering coefficient. Typically, low characteristic path length and high clustering coefficient are, however, conflicting characteristics in sparse graphs.

On the one hand, random graphs (e.g., Erdős-Rényi random graphs and generalized random graphs with arbitrary degree distributions) have a characteristic path length proportional to the logarithm of the number of vertices [10], but at the same time they have a low clustering coefficient. On the other hand, graphs

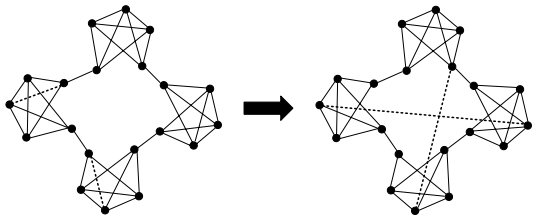


Figure 2: Peers forming a "Connected-caveman" graph, one of the most clustered connected graphs. The dashed lines show the edges that are rewired.

formed based on proximity criteria tend to exhibit a high clustering coefficient, but they have a high characteristic path length compared to random graphs [19]. These graph theoretic results are in accordance with results on the relationship between locality-aware overlay construction and streaming performance: locality-aware overlay construction leads to worse performance compared to random overlay construction [7].

There is, however, one class of graphs that satisfies both criteria of proximity-aware overlays. Small-world graphs, analyzed by Watts et al. [11], exhibit a high clustering coefficient but at the same time they have a characteristic path length comparable to that of random graphs of the same size.

There are numerous algorithms for creating small-world graphs (e.g., [20, 21]). All algorithms start from a highly clustered graph, whose edges are rewired in a probabilistic way, in order for it to be transformed into a small-world graph. The rewiring process consists of removing local edges from a cluster of vertices and replacing them with long distance edges to other clusters. Depending on the rewiring process, the same original clustered graph can be transformed into different small-world graphs with different characteristic path lengths.

A particular algorithm to construct a small-world graph starts from the so-called "connected caveman" graph [19], shown in Fig. 2. This graph is created from a number of isolated cliques by removing one edge from each clique, and using the removed edge to connect two neighboring cliques such that the connected cliques form a loop. During rewiring edges are taken one-by-one from the cliques and are rewired at random to another clique, as shown by the dashed lines in Fig. 2.

Given a flat ISP-level Internet topology with large connected components measurement data [12, 13], a proximity-aware overlay could be constructed to resemble small-world graphs. Each cluster of vertices corresponds to the peers in an ISP. Each cluster of vertices (peers in an ISP) has some connections to neighboring clusters, as defined by the ISP-level peering topology. During rewiring, every cluster of vertices (peers in an ISP) establishes a small number of connections with some appropriately chosen clusters of vertices (peers in other ISPs). The resulting overlay has a low characteristic path length, but at the same time many local connections

between peers in the same ISP.

4 Problem formulation and methodology

The algorithms used to create small-world graphs are based on a rewiring process that starts from a highly clustered graph. In the case of proximity-aware overlay construction over a flat ISP-level network topology the analogous to the rewiring process is the establishment of transit connections in an overlay that initially consists of local and peering connections.

Thus our focus is on understanding the impact of the establishment of transit overlay connections on the P2P system performance as a function of the underlying AS level network topology. The starting point of our investigation is a highly-clustered caveman-like overlay, which consists of local and peering overlay connections only. In this overlay every peer opens k connections to randomly chosen peers within the local ISP. Apart from the local connections, the peers in ISP i open in total π outgoing peering connections to the peers in each of the peering ISPs of ISP i, that is, to ISPs j for which $(i, j) \in E$. Finally, we allow the peers in every ISP i to establish in *total* τ outgoing transit connections to peers in non-peering ISPs.

The important question that we aim to answer is then the following. Given an ISP-level network topology, an overlay consisting of local and peering connections only, and a budget τ in terms of transit connections per ISP, how should the transit connections in the overlay be established in order to maximize the streaming performance. Answering this question is an important step towards engineering proximity-aware overlays for P2P streaming over a flat network topology.

4.1 Transit connection establishment policies

We consider transit connection establishment policies that rely on knowledge of the characteristics of the ISP-level network topology. In particular, we consider that every ISP knows its distance to the other ISPs in G and communicate this information to the overlay. The assumption is motivated by the ongoing standardization of services like ALTO [2].

The distance information is used by the overlay to choose the destination ISP of the transit overlay connections. The policies we consider are all special cases of the *Random-Distance Uniform AS* policy defined in the following.

Definition 1 (Random-Distance Uniform AS Policy) *Consider the graph G, and for every vertex (ISP) i of G a discrete distribution F_i given by its probability mass function $f_i(\delta)$ with domain $\{2, \ldots, max_j d(i, j)\}$. To establish a transit connection, a peer in ISP i picks a distance δ at random according to the distribution F_i. Then, among all ISPs j for which $d(i, j) = \delta$ the peer in ISP i chooses one ISP uniform at random, and establishes a transit connection to a peer in the chosen ISP.*

The Random-Distance Uniform AS policy allows transit connections to be chosen in different ways in different ISPs. Thus, the transit connection establishment policy used by peers in an ISP can be adapted to the location of the ISP in the AS-level graph G. This turns out to be important when the topology is asymmetric. In the following we introduce the transit connection establishment policies considered in the paper.

Deterministic Distance (*DetDist(η)*): The probability mass function $f_i(\delta)$ used by peers in ISP i is

$$f_i(\delta) = \begin{cases} 1 & \text{if } \delta = min(\eta, max_j d(i,j)) \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

Using this policy, the destination ISP of a transit connection established by peers in ISP i is choosen uniform at random among the ISPs that are η hops away, if there are such ISPs. Otherwise, the destination ISP is chosen uniform at random among the ISPs that are furthest away, that is, the destination is chosen uniform at random from the set $\{j : d(i,j) = min(\eta, max_j d(i,j))\}$. The *DetDist* policy allows precise control of the distance at which transit connections are established.

Binomially Distributed Distance (*BinDist(p)*): The probability mass function $f_i(\delta)$ used by peers in ISP i is

$$f_i(\delta) = \begin{pmatrix} max_j d(i,j) - 2 \\ \delta - 2 \end{pmatrix} p^{\delta-2}(1-p)^{max_j d(i,j)-\delta}. \quad (2)$$

Using this policy, the distance of the destination ISP of a transit connection established by peers in ISP i is chosen according to a binomial distribution with parameters $(max_j d(i,j) - 2,\ p)$, where $(i,j) \in E$. Through the probability p we can tune the average distance at which transit connections are established, but the control is not as precise as using the *DetDist(η)* policy.

Uniform Distance (*UnifDist*): The probability mass function $f_i(\delta)$ used by peers in ISP i is

$$f_i(\delta) = \frac{1}{max_j d(i,j) - 1}. \quad (3)$$

Using this policy, the distance δ for a transit connection established by peers in ISP i is chosen uniform at random from $\{2, \ldots, max_j d(i,j)\}$. Compared to *BinDist(p)*, using *UnifDist* the distance of the ISP to which a connection is established is more spread.

Uniform AS (*UnifAS*): The probability mass function $f_i(\delta)$ used by peers in ISP i is

$$f_i(\delta) = \frac{|\{j | d(i,j) = \delta\}|}{\sum_{\delta'} |\{j | d(i,j) = \delta'\}|}. \quad (4)$$

Using this policy, the destination ISP j of a transit connection established by peers in ISP i is chosen uniform at random among all the non-peering ISPs, i.e., ISPs that are at least at distance $d(i,j) = 2$. Since the distance distribution is induced

by the AS topology, it is in general *not* uniform. The advantage of this policy is that it can be implemented by knowing only the set of peering ISPs and the set of all ISPs, i.e., *without* knowing the complete AS topology. We use this policy as a baseline.

Inverse Distance Proportional (*InvDist*): The probability mass function $f_i(\delta)$ used by peers in ISP i is

$$f_i(\delta) = \frac{|\{j|d(i,j) = \delta\}|/\delta}{\sum_{\delta'} |\{j|d(i,j) = \delta'\}|/\delta'}. \quad (5)$$

Using this policy, the destination ISP j of a transit connection established by peers in ISP i is chosen with a probability inverse proportional to the distance of j from i. We consider this policy, because it was shown to be optimal for the routing of a message from a source to a destination based only on local knowledge [22].

4.2 Evaluation Methodology

We use extensive simulations to evaluate the performance of the different transit connection establishment policies. The simulations were performed in P2PTV-SIM [23], a packet level event-driven simulator. We simulated a static overlay, constructed according to one of the policies presented in Section 4.1, after all peers joined. As mesh-based P2P streaming systems are known to be robust to node churn, we expect that simulating churn would not significantly affect our conclusions [24, 25].

After the overlay is constructed, the source peer starts the streaming of chunks. The chunks are distributed by the peers until their playout deadline B is reached. Data forwarding is based on a random push algorithm [24, 25]. A peer forwards to a random neighbor a randomly chosen chunk out of the chunks that the neighbor is missing. After a chunk has been scheduled for transmission to a peer, no neighbor of the recipient peer will schedule the same chunk, avoiding thus the reception of duplicate chunks and the resulting waste of upload capacity. We quantify the streaming performance using the chunk *missing ratio*, which is defined as the number of chunks not received by their playout deadline divided by the total number of chunks. We do not consider losses in the network layer, therefore the missing chunks are solely attributed to the P2P protocol, that is, the overlay topology and the forwarding algorithm.

We quantify the available upload capacity in the system by the resource index (RI), which is the ratio of the aggregate peer upload capacity over the aggregate download capacity demand of the peers. The aggregate download capacity demand of the peers is the video streaming rate times the number of peers. The video is segmented into chunks of 50 kbits and is streamed by the source server at a constant rate of 500 kbps. To account for the effect of propagation delays on the chunk diffusion we use one-way network latencies of 10 ms for local connections and of 25 ms for peering and for transit connections. Unless otherwise stated, in our

simulation setup there are 40 peers in each ISP, each of which has $k = 7$ outgoing local connections in total. The playout deadline for chunks is set to $B = 6$ seconds.

5 The case of a ring ISP topology

We first consider the case that the graph G formed by the peering relations between ISPs is a ring. In this case every ISP has peering links with 2 ISPs. Without loss of generality we denote the ISP in which the streaming server resides by ISP 0. The streaming server forwards data exclusively to peers within ISP 0. In the following, we present simulation results for the construction policies over a ring of 30 ISPs.

In Figure 3 we show the chunk missing ratio versus the number of peering connections for an overlay with $RI = 1.3$ and the *BinDist*, *UnifAS*, and *InvDist* policies for one transit connection per ISP ($\tau = 1$). We do not show results for the *UnifDist* policy, because on a ring topology it is equivalent to the *UnifAS* policy. Instead, we show results for an overlay that consists of local and peering connections only ($\tau = 0$). As expected, there is a significant difference between the results without ($\tau = 0$) and with ($\tau = 1$) transit connections. What is surprising is that the difference between the worst (*BinDist(0.1)*) and the best (*BinDist(0.75)*) transit connection policy is actually much bigger than that between the results without transit connections ($\tau = 0$) and the worst transit connection establishment policy (*BinDist(0.1)*). Furthermore, the difference between the results for the different transit connection establishment policies increases as the number of peering connections π increases. The worst performance is achieved by the *BinDist(0.1)* policy, which tends to establish transit connections between peers in nearby ISPs. The policies that favor transit connections between peers in distant ISPs yield consistently better results, this phenomenon can be well observed on the performance of the *BinDist(p)* policies, which increasingly favor transit connections between peers in distant ISPs as p increases.

5.1 Does Distance Matter?

The fact that the *BinDist(p)* policies with large p values show better performance suggests that it is favorable to open transit connections to distant ISPs. In the following we use the *DetDist(η)* policies to verify this hypothesis. To compare the *DetDist(η)* and the *UnifAS* policies, we define the *relative gain* $\rho(\eta)$ for $p_{loss}(UnifAS) > 0$ as

$$\rho(\eta) = \frac{p_{loss}(UnifAS) - p_{loss}(DetDist(\eta))}{p_{loss}(UnifAS)},$$

where $p_{loss}(\Pi)$ is the chunk missing ratio for policy Π, measured over the same overlay.

In Figure 4 we show the relative gain for three overlays with different number of peering connections. We show results for distances $\eta \geq 6$, because for lower values of

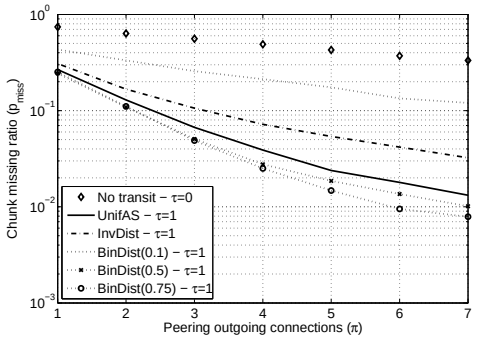


Figure 3: Chunk missing ratio vs π for various policies. Overlay with RI=1.3. Ring of 30 ISPs.

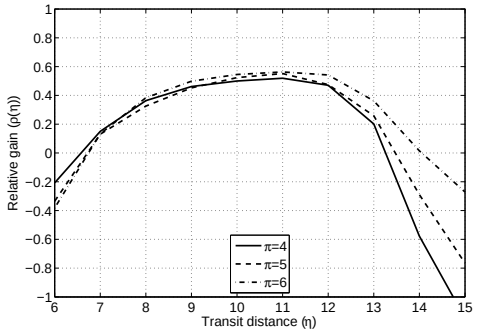


Figure 4: Relative gain ($\rho(\eta)$) of the *DetDist(η)* policy over the *UnifAS* policy. Overlay with $\tau = 1$. Ring of 30 ISPs.

η the performance is very poor compared to the *UnifAS* policy. We observe a wide range of η values for which the performance of the *DetDist(η)* policy is significantly better than that of the *UnifAS* policy. For all three curves the maximum gain is achieved at $\eta = |I|/3$ approximately, where $|I|$ is the length of the ring of ISPs. The maximum gain exceeds 50 percent, which would be equivalent to a $3dB$ decrease of video distortion according to recent loss-distortion models of scalable and non-scalable video [26].

Intuitively, one would expect that establishing transit connections between peers in distant ISPs decreases the characteristic path length of the overlay, which is the reason for the improved streaming performance. To test this hypothesis, in Figure 5 we show the characteristic path length of the overlay for the considered policies. The characteristic path length changes significantly as a function of η for *DetDist(η)*,

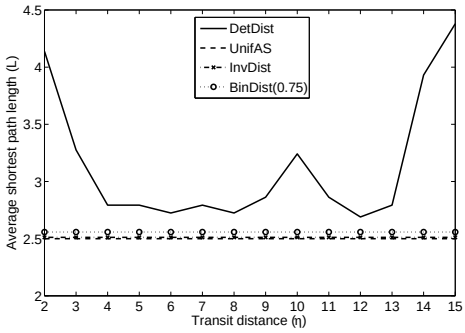


Figure 5: Shortest path lengths from the source ISP for the *UnifAS*, *DetDist* and *InvDist* policies. Ring topology of 30 ISPs.

and is always higher than for the *UnifAS* and *InvDist* policies. For the smallest and highest values of η the characteristic path length is significantly higher than for the *UnifAS* and for the *InvDist* policies, which could potentially explain the higher chunk missing ratios for *DetDist(η)* at these values of η. For values of η between 4 and 14 this reasoning seems to fail. In fact some of the highest gains appear at η values (e.g., $\eta = 10$ and $\eta = 14$) where the average path length reaches a local maximum. Nevertheless, using *DetDist(η)* the lowest chunk missing ratio is achieved for the value of η for which the characteristic path length is minimal. We made similar observations on ring topologies of different sizes.

The evaluation of the distance-based policies shows the importance of having short path lengths for the performance of the streaming system. However, despite the fact that shortest paths are essential, results from the *DetDist(η)* policy reveal that the system performance depends also on how the shortest paths are constructed. We are, thus, led to the hypothesis that the diffusion efficiency depends as well on the ISP disjointness of the paths that are created by the transit connection establishment policies. Disjoint paths between the ISP hosting the source server and the rest of the ISPs allow the load of chunk forwarding to be shared between peers in many ISPs. Thus, the data diffusion becomes faster and more efficient in terms of resource utilization.

5.2 Per-ISP Loss Statistics

To give further insight into the impact of the transit connection establishment policies on the streaming performance, in Figure 6 we show the histogram of the chunk missing ratio of the peers averaged per ISP for the *UnifAS*, *DetDist(11)* and *DetDist(13)* policies. In the legend we show the average, the maximum and the coefficient of variation (CoV, the ratio of the standard deviation and the mean) of

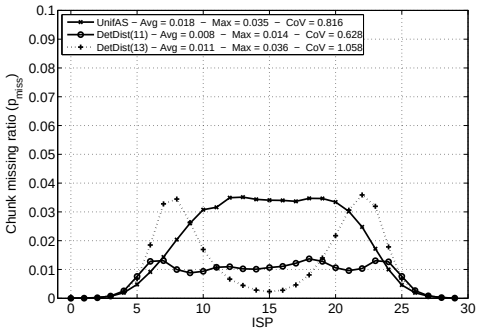


Figure 6: Probability mass function of chunk missing ratio experienced by the peers in each of the ISPs. The legend shows the average, maximum chunk missing ratio among all ISPs and the standard deviation of the per ISP losses.

the per ISP chunk missing ratios. The different transit connection establishment policies lead to distributions with very different characteristics: the *UnifAS* policy leads to a bell-shaped distribution with highest losses in the ISPs that are furthest away from ISP 0, the *DetDist(11)* policy leads to a rather uniform distribution, and the *DetDist(14)* policy leads to a bimodal, almost symmetric distribution. While the *DetDist(11)* policy outperforms the other policies in terms of all three metrics (average, maximum and CoV), comparing the other two policies is problematic: the *DetDist(13)* policy has lower average chunk missing ratio than the *UnifAS* policy but higher CoV.

Unfortunately, the average chunk missing ratio tells little about the chunk missing ratios in the different ISPs, as the difference between the different ISPs is more than an order of magnitude for each of the three policies. Hence, in general, the average chunk missing ratio calculated over all peers might not be a sufficient metric for the performance evaluation of proximity-aware overlays. The large difference in terms of chunk missing ratios between ISPs becomes very important when video coding or channel coding parameters are optimized for the average calculated over all peers in the overlay [27]. In Section 7 we show that this observation does not only hold on a ring ISP topology, but it holds also when using a real ISP-level topology.

6 The case of a square grid ISP topology

The second regular graph topology we consider is a two-dimensional square grid of ISPs connected by peering links. ISPs in the interior of the grid have peering links with 4 ISPs, the ISPs that are on the edges of the grid have peering links

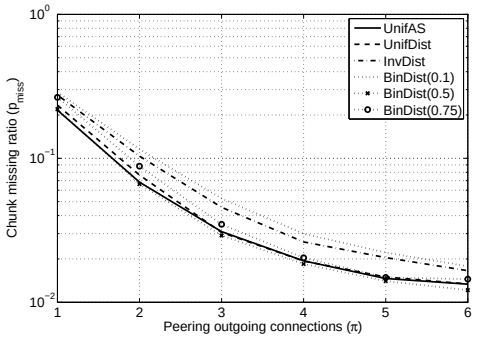


Figure 7: Chunk missing ratio vs number of peering connections for $\tau = 1$. Server ISP in the middle of a 11×11 grid.

with 3 ISPs and the ISPs at the grid corners have peering links with 2 ISPs. On the square grid topology the inter-ISP distance distributions are non-homogeneous, which allows us to explore the impact of the server placement on the policies.

We first consider the case when the streaming server is placed in the ISP at the middle of the grid. Presumably, placing the server in the middle minimizes the average number of ISP hops that data have to travel in the overlay. In the following we present results for a grid topology of 11×11 ISPs.

In Figure 7 we show the chunk missing ratio as a function of the number of peering connections π for various policies. For a few peering connections the policies show similar performance. Nevertheless, as the number π of peering connections between ISPs increases the *InvDist* and the *BinDist(0.1)* policies perform significantly worse than the rest. Among the other policies, *BinDist(0.5)* and *UnifAS* perform best for all values of π, and *UnifDist* performs almost equally well. Thus, again, transit connections between peers in distant ISPs provide the best performance.

6.1 Streaming server placement

To explore the impact of the location of the streaming server on the relative performance of our policies, we now consider the case when the server is located in the ISP at a corner of the grid. In Figure 8 we show the chunk missing ratio versus the number of peering connections for all the considered policies. The difference between the results obtained with the different policies is significantly higher than when the server is placed in the middle (Fig. 7). Still, the policies that favour the short distance transit connections fare worse, while the *UnifDist* and *BinDist* policies with $p = 0.5$ and $p = 0.75$ fare best. Interestingly, changing the location of the streaming server negatively affected the performance of the *UnifAS* policy,

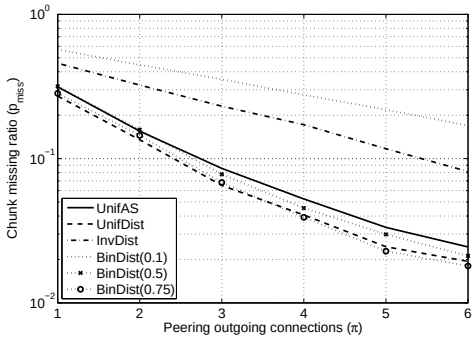


Figure 8: Chunk missing ratio vs. number of peering connections for $\tau = 1$. Server ISP in the upper left corner of the 11×11 grid.

but not that of the *UnifDist* policy. We explain this phenomenon by that for the ISPs close to the grid's corner the *UnifDist* policy establishes transit connections to peers in distant ISPs at a higher probability than the *UnifAS* policy. The significant difference between the results obtained for the two server locations shows that the optimal policy for establishing transit connections is not only a function of the ISP-level network topology, but also of the location of the server.

6.2 Topology scaling

The next question we look at is the impact of grid topology size on the performance of the policies. In Fig. 9 we show the relative gain (ρ) of the *UnifDist* and *BinDist* policies over the *UnifAS* policy versus the size of the grid topology for $\pi = 3$ peering and $\tau = 1$ transit connections. The server is placed in the upper left corner of the grid. The gain for the *BinDist* policies that favour long distances is significant, it exceeds 40% for the 7×7 grid. Nevertheless, their gain decreases almost linearly proportional to the grid size. Contrary to the *BinDist* policy, the gain of the *UnifDist* policy seems to be insensitive to the size of the grid.

7 The case of the Internet AS topology

The last topology we consider is based on the Internet AS-level peering topology in the CAIDA dataset [28]. The dataset contains 36878 ASs and 103485 links between ASs. The inter-AS links are characterized as transit provider-customer and peering. The inference of the recorded relationships is based on methodology described and validated in [13]. We focus on the set of ASs that have at least one peering link. The set consists of 1200 ASs, and these 1200 ASs form 83 connected components.

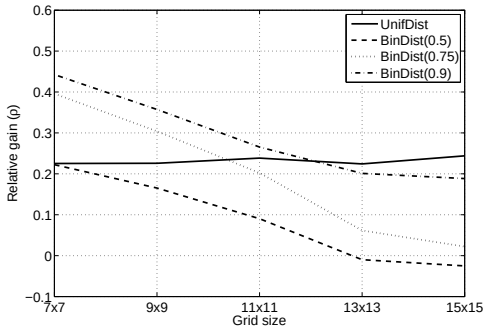


Figure 9: Relative gain of *UnifDist* and *BinDist* policies as a function of the grid size. Overlay with $\pi = 3$ and $\tau = 1$. Server ISP in the upper left corner of the grids.

The largest connected component consists of 616 ASs connected by peering links. We refer to this connected component as the *Caida-Peer* graph, and we use this connected component as the graph G defined in Section 3.

The characteristics of the *Caida-Peer* graph differ significantly from those of the AS-level topology in the complete CAIDA dataset including transit connections. In Figure 10(a) we show the rank-degree statistics for the complete dataset and for the *Caida-Peer* graph. The AS rank-degree statistics for the complete CAIDA dataset follows a power-law over a range that spans 2 orders of magnitude of the AS rank. Accordingly, the average distance between the ASes is small, only 3.8 hops. The rank-degree statistics of the *Caida-Peer* graph does not exhibit a power-law, and the maximum AS degree is more than an order of magnitude lower than for the complete CAIDA topology. Despite the lower number of ASs in the graph, the average AS distance is higher, 4.57 hops, and the diameter of the graph is 12 hops.

Figure 10(b) shows the cumulative distribution function of the average shortest paths for the *Caida-Peer* graph and for the 7×7 grid topology used in the previous section. Interestingly, both topologies have a diameter of 12 hops and almost the same median and average distance distance between ASes. Given the similarities of the graphs' characteristics, an intriguing question is whether the relative performance of the different policies on the two topologies is also similar.

To investigate this question we consider overlays with 20 peers in each of the ASs of the *Caida-Peer* graph, homogeneous upload capacities and a resource index of $RI = 1.25$. Figure 11 shows results obtained using the three policies that showed the best performance in the previous sections: *UnifAS*, *BinDist* and *UnifDist*. Motivated by the importance of the server placement observed for the grid topology, we consider two server placements: (a) in the AS with the highest degree, (b) in the

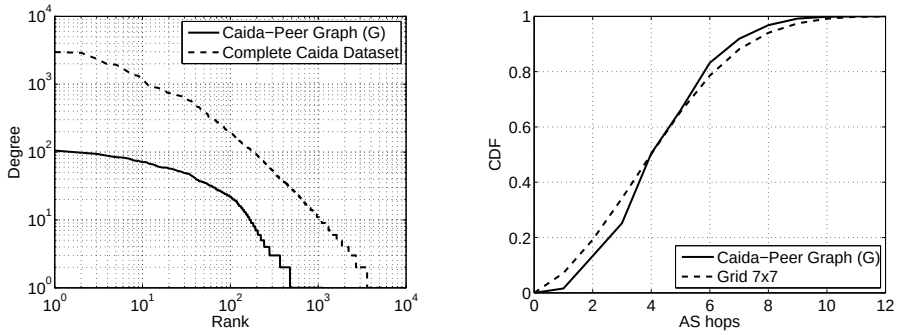
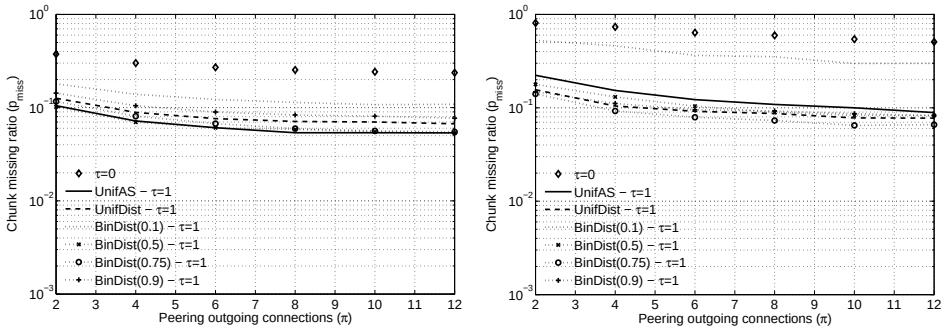


Figure 10: Characteristics of the *Caida-Peer* graph: (a) Rank-degree plot of the CAIDA dataset and the *Caida-Peer* graph and (b) Comparison of the CDF of the average distance between ASes for the grid of 7×7 and the *Caida-Peer* graph

AS with the lowest degree and the highest distance from all other ASs. The results obtained for these two server placements show very similar characteristics to the results obtained on the 11x11 grid topology for the two different server placements (Figs 7 and 8). First, the performance of all policies is better when the server is placed in a centrally located AS. Second, the relative performance of the different policies is very similar. The *UnifAS* policy performs best when the server is placed in the AS with the highest degree, but is outperformed by the *UnifDist* and several of the *BinDist* policies when the server is placed in the AS with the lowest degree. The similarity of the conclusions drawn based on results for the grid and the *Caida-Peer* graph shows that the overlay construction policy should be chosen as a function of the location of the server to maximize the streaming performance.

Figure 12 shows the cumulative distribution function of the loss ratio in the ASs of the *Caida-Peer* graph for $\pi = 6$, without transit connections ($\tau = 0$) and for three transit connection establishment policies ($\tau = 1$).

Without transit connections the loss ratio is prohibitively high for streaming: the average loss ratio is 0.27, and it is higher than 0.1 in 50 percent of the ASs. Allowing for one outgoing transit connection per AS decreases the average loss ratio by almost up to an order of magnitude (0.12, 0.067 and 0.06 for the *BinDist(0.1)*, *BinDist(0.75)* and *UnifAS* policies, respectively). Furthermore, the peers in less than 15% of the ASs have a loss ratio above 0.2. Still, the difference between the loss ratios in the ASs is very high, which confirms the observation made on the ring topology that the average loss ratio is not a reliable indicator of the system performance for network-aware overlays.



(a) Server placed in highest degree AS

(b) Server placed in most remote AS

Figure 11: Chunk missing ratio vs number of peering connections on the *Caida-Peer* graph.

7.1 Non-uniform peer populations

To study the sensitivity of the results to the distribution of the peers in the ASs, in the following we consider an overlay in which the number of peers per AS follows a shifted Mandelbrot-Zipf distribution with parameters $\alpha = 1.33$, $q = 10$ and a shift of 10 peers. These parameters were obtained by fitting recent measurement data [29]. The peers were assigned to the ASs of the *Caida-Peer* graph according to the rank of the ASs in terms of their degree, motivated by that ASs with many peering links are likely to belong to ISPs with a large customer base and have thus higher probability of hosting more peers than poorly connected ASs.

In Fig. 13 we show the chunk missing ratio as a function of the number of peering connections π for two server placements: (a) in the AS with the highest degree, (b) in the AS with the lowest degree and the highest distance from all other ASs. Our first observation is that the average chunk missing ratio is an order of magnitude smaller than for the uniform peer distribution over the ASs (Fig. 11). The reason is that by levaraging the peering links between the ASs and the power-law distribution of the peers in the ASs, we achieve a good match between the overlay and the network topology. Our second observation is that the relative performance of the policies is similar to that for the case of the uniform peer population, and indicates that the results are not very sensitive to the distribution of the peers over the ASs.

7.2 Non-uniform peer bandwidths

Finally, we study the sensitivity of the results to the upload bandwidth distribution of the peers. We consider a scenario where peers belong to four classes with respect to their upload bandwidth as shown in Table 1. The average upload bandwidth in the overlay is 1.25 Mbps. We maintain the resource index (RI) in the overlay

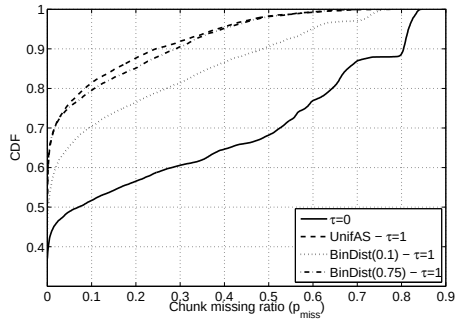


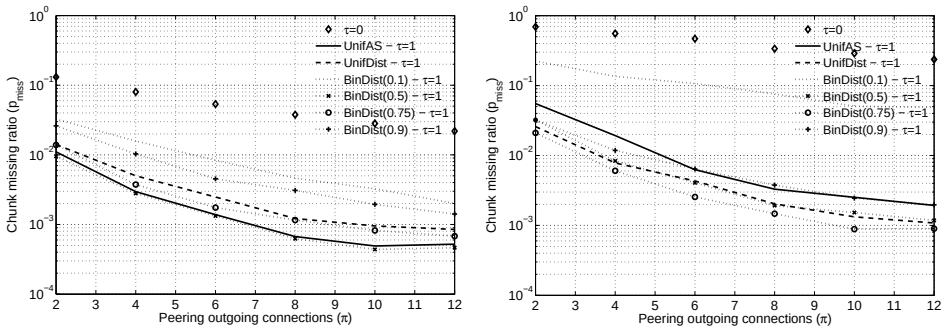
Figure 12: Cumulative distribution function of the loss ratio in the ASs of the *Caida-Peer* graph. Overlay with $\pi = 6$.

to $RI = 1.25$ as with the homogeneous peer upload bandwidth distribution and consequently the video rate is set to 1 Mbps and the chunk size to 100 kbits. The peers are distributed over the ASs according to the shifted Mandelbrot-Zipf distribution as in the previous sub-section.

Class	Upload (Mbps)	Peers (%)
1	5	10
2	1.5	10
3	1	40
4	0.55	40

Table 1: Peer Bandwidth Classes

In Fig. 14 we show the chunk missing ratio as a function of the number of peering connections for the policies that performed best in the previous sections and for two server placements: in the highest degree AS and in the AS with the lowest degree and the highest average distance from all other ASs. When the server is placed in the AS with the highest degree, all three policies have almost identical performance, as was the case for the homogeneous bandwidth scenario. Similarly, when the server is placed in the AS at the highest distance, we observe that the *BinDist(0.75)* and the *UnifDist* policies perform distinctively better than the *UnifAS* policy for all π values, confirming our previous results. In comparison to the results for the overlay consisting of peers with homogeneous upload bandwidths (Fig. 13(b)), we see that the heterogeneous overlay performs significantly worse, exhibiting up to an order of magnitude higher chunk missing ratio for all the transit connection establishment policies. Nevertheless, the better performance of the *BinDist(0.75)* and *UnifDist* policies is observed in both scenarios, which shows that our results are not sensitive



(a) Server placed in highest degree AS

(b) Server placed in most remote AS

Figure 13: Chunk missing ratio vs number of peering connections. Peers distributed to ASs according to the Mandelbrot-Zipf distribution.

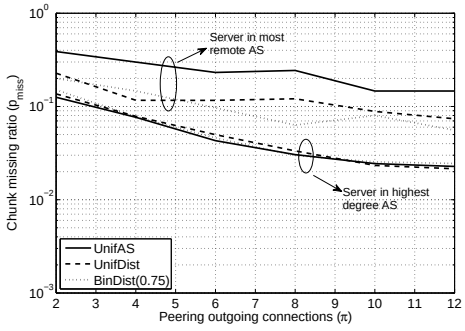


Figure 14: Chunk missing ratio vs. number of peering connections for non-uniform upload bandwidths. Peers distributed to ASs according to the Mandelbrot-Zipf distribution.

to the upload bandwidth distribution of the peers under random chunk forwarding.

8 Conclusion

We considered the problem of network-aware overlay construction over a network of ISPs with peering relationships. Motivated by the analogies between the construction of small-world graphs and the construction of proximity-aware overlays, we proposed several topology-aware policies to establish transit connections between peers in remote ISPs. We used extensive simulations over various ISP topologies, among them the measured peering topology of over 600 autonomous systems, to compare the policies, and to get insight into the interplay between network topology, overlay topology and streaming performance. Our results show that by adapting the connection establishment policy to the network topology the performance of a network-aware overlay can be improved significantly. In general, transit connections should be opened between distant ISPs, and choosing uniform at random often, but not always, implements such a policy. We showed that besides the network topology itself, the location of the streaming server in the network topology is an important factor that affects the performance of the overlay construction policies. Finally, we also pointed out that the average loss probability is often not a sufficient performance metric to assess the performance of proximity-aware overlays.

References

[1] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *Proc. of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 6–6, 2005.

[2] IETF ALTO Working Group, https://datatracker.ietf.org/wg/alto/charter/.

[3] D. Choffnes and F. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *Proc. of ACM SIGCOMM*, 2008.

[4] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proc. of the International Conference on Distributed Computer Systems (ICDCS)*, 2006.

[5] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can isps and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37:29–40, July 2007.

[6] S. Le Blond, A. Legout, and W. Dabbous. Pushing bittorrent locality to the limit. *Computer Networks*, 2010.

[7] F. Picconi and L. Massoulié. ISP-friend or foe? making P2P live streaming ISP-aware. In *Proc. of the 29thIEEE International Conference on Distributed Computer Systems (ICDCS)*, pages 413–422, 2009.

[8] Xin Jin and Yu-Kwong Kwok. Network aware P2P multimedia streaming: Capacity or locality? In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 54 –63, Sept. 2011.

[9] J. Seedorf, S. Niccolini, M. Stiemerling, E. Ferranti, and R. Winter. Quantifying operational cost-savings through alto-guidance for P2P live streaming. In *3rd Workshop on Economic Traffic Management (ETM 2010)*, September 2010.

[10] B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.

[11] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[12] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proc.of ACM SIGCOMM*, pages 75–86, 2010.

[13] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, Kc Claffy, and G. Riley. AS Relationships: Inference and Validation. *SIGCOMM Comput. Commun. Rev.*, 37:29–40, January 2007.

[14] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4P: Provider portal for applications. In *Proc. of the ACM SIGCOMM 2008*, pages 351–362, 2008.

[15] D. Ciullo, M. A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia. Network awareness of P2P live streaming applications: a measurement study. *IEEE Transactions on Multimedia*, 12:54–63, January 2010.

[16] Gy. Dán. Cooperative caching and relaying strategies for peer-to-peer content delivery. In *Proc. of Intl Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2008.

[17] J. Dai, B. Li, F. Liu, B. Li, and H. Jin. On the efficiency of collaborative caching in isp-aware P2P networks. In *Proc. of IEEE INFOCOM*, Apr. 2011.

[18] A. Dhamdhere and C. Dovrolis. The Internet is flat: modeling the transition from a transit hierarchy to a peering mesh. In *Proc. of ACM Co-NEXT*, pages 1–12, 2010.

[19] D. J. Watts. *Small worlds: The dynamics of networks between order and randomness*. Princeton University Press, 1999.

[20] M.E.J. Newman, C. Moore, and D. J. Watts. Mean-field solution of the small-world network model. *Physical Review Letters*, 84:3201–3204, 2000.

[21] S. N. Dorogovtsev and J. F. F. Mendes. Exactly solvable analogy of small-world networks. *EUROPHYS.LETT.*, 50:1, 2000.

[22] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symposium on Theory of Computing*, 2000.

[23] P2PTV-SIM, http://napa-wine.eu/cgi-bin/twiki/view/public/p2ptvsim.

[24] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: Optimal performance trade-offs. In *Proc. of ACM SIGMETRICS*, 2008.

[25] I. Chatzidrossos, Gy. Dán, and V. Fodor. Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates. *P2P Networking and Applications*, 3:208–221, March 2010.

[26] V. Vukadinovic and Gy. Dán. Multicast scheduling for scalable video streaming in wireless networks. In *ACM Multimedia Systems (MMSys)*, February 2010.

[27] Gy. Dán and V. Fodor. Stability and performance of overlay multicast systems employing forward error correction. *Performance Evaluation*, 67:80–101, February 2010.

[28] CAIDA. Automated Autonomous System (AS) Ranking, http://as-rank.caida.org/data/.

[29] Haiyang Wang, Jiangchuan Liu, and Ke Xu. On the locality of BitTorrent-based video file swarming. In *Proceedings of the 8th international conference on Peer-to-peer systems*, IPTPS'09, Berkeley, CA, USA, 2009.