Multimedia
(will be inserted by the editor)

Enabling Resilient P2P Video Streaming: Survey and Analysis

Osama Abboud · Konstantin Pussep
Aleksandra Kovacevic · Katharina Mohr
Sebastian Kaune · Ralf Steinmetz

Received: date / Accepted: date

Abstract Peer-to-Peer (P2P) techniques for multimedia streaming have been shown to be a good enhancement to the traditional client/server methods when trying to reduce costs and increase robustness. Due to the fact that P2P systems are highly dynamic, the main challenge that has to be addressed remains supporting the general resilience of the system. Various challenges arise when building a resilient P2P streaming system, such as network failures and system dynamics. In this paper, we first classify the different challenges that face P2P streaming and then present and analyze the possible countermeasures. We classify resilience mechanisms as either core mechanisms, which are part of the system, or as cross-layer mechanisms that use information from different communication layers, which might inflict additional costs. We analyze and present resilience mechanisms from an engineering point of view, such that a system engineer can use our analysis as a guide to build a resilient P2P streaming system with different mechanisms and for various application scenarios.

1 Introduction

Peer-to-Peer (P2P) video streaming is becoming increasingly popular as it not only helps in reducing costs and flash crowd sensitivity, but also allows for new innovative applications such as social networking and support for user-generated content. Therefore, a vast variety of P2P video streaming systems have been developed. Video streaming applications include both client-server based approaches, such as YouTube[1], and P2P-based appraoches, such as Zattoo[2], PPLive[3], Octoshape[4], and CoolStreaming [1] to name a few.

Traditional client-server based technologies offer the possibility to provide good performance and high availability rates. However, those approaches usually inflict high deployment and maintenance costs. The current estimation of YouTube's costs is 1 million dollars per day [2] and these costs could increase drastically if more videos continue to be switched to higher qualities. Therefore, various P2P streaming systems have been proposed to enable live and Video-on-Demand (VoD) streaming to large audiences with better video quality at low deployment and server costs [3]. Furthermore, utilization of the P2P paradigm yields several advantages, such as inherent bandwidth and resource scalability, network path redundancy, and self organization [4,5].

Unfortunately, these advantageous properties come at a cost. P2P systems are dynamic and heterogenous by nature and, therefore, introduce many challenges and problems, especially for resilience. Peers with weak resources are usually unable to actively participate in systems, as those systems usually apply a static selection of streaming parameters based on average peer resources [6]. Furthermore, a lack of coordination makes the deployment of efficient P2P streaming systems challenging. This problem is worsened due to low system stability and churn, which refers to peers joining and leaving the network unpredictably.

Due to these issues, the deployment of mechanisms to overcome the effects that errors and failures have on the streaming quality, is an exigency. This survey aims

Multimedia Communications Lab, Technische Universitaet Darmstadt, Rundeturmstr. 10, 64283 Darmstadt, Germany
E-mail: {abboud, pussep, sandra, mohr, kaune, steinmetz}@kom.tu-darmstadt.de

[1] YouTube, http://www.youtube.com
[2] Zattoo, http://www.zattoo.com
[3] PPLive, http://www.pplive.com
[4] Octoshape, http://www.octoshape.com

at providing an overview of the problems that current deployments face, and how resilience against various types of problems can be achieved. The main contribution of this paper is that it can be used as a reference by system developers. Therefore, a resilient P2P streaming system can be developed by choosing resilience mechanisms from the choices that we present, analyze, and discuss in this paper.

The rest of the paper is organized as follows. Section 2 provides background knowledge on P2P video streaming, then in Section 3 we define resilience with a discussion on the different challenges that arise when building a resilient P2P streaming system. Resilience mechanisms are classified either as core or cross-layer mechanisms. The first class is described in Section 4 with emphasis on the different possible overlay topologies. The second class of mechanisms is presented in Section 5 with discussions on how to utilize information from different communication layers to enhance the resilience of the system. Finally, in Section 6 we present a discussion and conclusion on the presented mechanisms and their combinations.

2 Background

Videos can be provided either in the download mode or in the streaming mode. While in the download mode, videos can be played only after they are completely downloaded, streaming refers to play-out during download. Therefore, streaming allows for lower waiting times, as playback can start as soon as the playback buffer is full. This, however, introduces stringent timing requirements for the streaming application. Since video pieces - which are small video transmission units - are not useful if they are not received on time, the network architecture and mechanisms must be carefully chosen in order to ensure that bandwidth, delay, and loss requirements are fulfilled.

The focus of this paper is on P2P systems, as they provide advantageous properties such as self organization and resource scalability. Other benefits are distributed storage space, a large amount of resources for computation, and redundancy of network paths. This all leads, with the adequate distributed techniques, to a performant and robust video delivery. High costs due to bandwidth needs at the server are a major drawback of client-server architectures, as this leads to extreme costs for scalability. With an increased number of clients the required bandwidth grows proportionally [3]. Therefore, P2P techniques aid the relaxation of bandwidth burdens placed on servers, as peers act both as providers and consumers, making it possible to provide low cost video streaming services [7].

3 Resilience

Resilience is not only a computing related term, but is also used in social psychology, material science as well as in ecology, business, and industrial safety. In all previously mentioned domains the general definition holds, namely the ability of effective accommodation to unpredictable environmental disturbances. Although resilience is mostly used interchangeable with fault tolerance, it can also be used to describe dependability and robustness. A definition of resilience is given in [8], namely: "The persistence of avoiding too frequent or severe failures in the presence of changes". Changes can, therefore, be classified by their nature, their predictability, and their timing. A system can be denoted as resilient, if it has the ability to sustain a specific level of service quality even in the face of a certain amount of failures.

Resilience is of significant importance for P2P streaming, since perceived media quality suffers noticeably upon failures [9]. In addition, P2P streaming applications are built on top of communication infrastructures and, therefore, failures in the underlay network also have an adverse impact on the performance of the streaming system. The term failure, in this context, includes complete failures such as a peer getting disconnected from the network, or partial failures such as a sudden drop in download throughput.

3.1 Challenges

Deployment of P2P-based media streaming systems entails several challenges. To start off, a main challenge is the required timing constraint. This stems from the fact that, unlike in file-sharing applications, video files are directly played-out while they are being downloaded [10]. Therefore, pieces, which are received after their play-out time, degrade user experience. This degradation is visible either as missing frames or as a playback stop, which is also denoted by stalling. Both loss and delay of pieces can cause this problem. While redundancy schemes might be suitable for streaming because they do not require further communication between sender and receiver, retransmission might not be possible, because of the strict timing requirements.

In addition, peers have limited upload capacities, which stems from the fact that the Internet (especially last mile connections) was designed for the client/server paradigm and applications. Additionally, peers are residing at the edge of the network, which leads to suboptimal utilization of P2P resources due to traffic aggregation and makes it difficult to utilize all of a peer's

download capacity [11]. Due to the limited upload capacities, it might not be possible for a single peer to utilize all of its download capacity for receiving a stream from a single peer. Therefore, a set of serving peers - that are dynamic - is required for video delivery, leading to a higher degree of dynamics in the system. Furthermore, peers are heterogenous with various classes of bandwidth capabilities, therefore, accommodation of streaming bit-rates for the whole topology is necessary. Differences in bandwidth can be static, caused by different link capacities, as well as dynamic, which can be network-induced.

Additionally, streaming systems suffer from packet drop or delay due to network congestions. While addressing these challenges, the proper choice of stable and reliable neighbors for delivery at the overlay plays a significant role. As churn might be frequent in streaming systems, it remains a challenging task to ensure that peers retain connectivity to be able to cope with a certain amount of failures.

3.2 What Affects Resilience of P2P Streaming Systems?

To better understand the challenges of achieving a resilient P2P streaming system, we classify those challenges according to where they occur in the communication layers. In the context of P2P systems, the relevant communication layers are: underlay, overlay, application, and user layers.

The most important layers in a P2P streaming system are the underlay and overlay. An overlay is defined as the virtual network that is constructed over the physical network (called the underlay). An underlay abstracts all lower layers in the traditional network layer model. An overlay layer could also be used to introduce new mechanisms to ensure resilience such as building different links between peers.

Table 1 shows how resilience is affected starting from the underlay, to the overlay, going through the application layer towards the user layer. Additionally, Figure 1 depicts the mentioned challenges within a sample network consisting of two connected Autonomous Systems (AS).

User	Free riding, malicious behavior
Application	Costs, buffering policy, strict timing requirements
Overlay	Peer failures, churn, neighbor selection
Underlay	Heterogeneity, limited resources, throughput, packet loss, packet delay, jitter, congestions

Table 1 How resilience is affected at different layers.

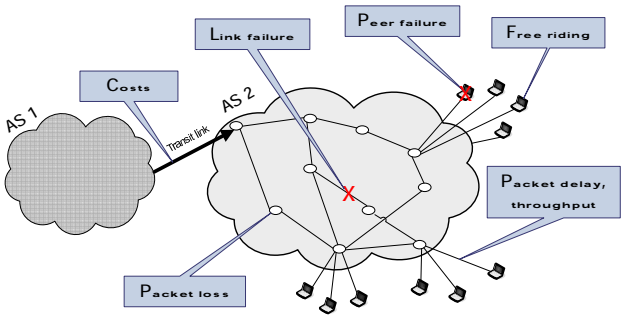

Fig. 1 Resilience of P2P video streaming systems can be affected by various aspects of different communication layers.

3.2.1 Underlay Layer

Just like classical communication networks, P2P systems suffer from structural failures at the network core of the underlay network. Performance, as perceived at the edge of the network, is characterized by certain throughput, packet loss, packet delay, and jitter. Problems at the underlay can result from breakdown of links, routers, or other neighboring peers. Then, data often cannot be transferred at all or has to take a different and more costly path from sender to receiver. Also congestion arises when the load is placed on the network exceeds the capacity of the network, leading to packet loss. Another issue for resilience at the underlay layer is resource limitation and heterogeneity. Nowadays, devices with various resources are joining P2P streaming systems. For these devices, resources such as upload and download bandwidth or storage space, can be limited or expensive. For example, the storage space might affect how many pieces can be buffered, whereas low bandwidth capacities might lead to pieces being dropped, while sending or receiving. This all puts restrictions on possible contribution from those peers.

3.2.2 Overlay Layer

Resilience in P2P systems is not only affected by the above mentioned issues at the underlay, but also by peer dynamics at the overlay. In P2P systems the network is very dynamic, therefore, the availability of content varies frequently as peers leave and join the network. Peers can depart from the overlay network gracefully, or they can fail, which also causes the breakdown of overlay links. Due to these network dynamics, re-routings might be necessary, leading to an increase in packet delay. Also the overlay network can become partitioned if a great number of links break down [9]. Furthermore, if neighbors are not selected properly, this can also have adverse effects on resilience. For example, the selected

neighbors can be unreliable or the distance between peers can be large, which causes an increase in required transmission time [12].

3.2.3 Application Layer

At the application layer, resilience can be affected by network aspects or the streaming policies. Network aspects are mainly costs of communication between different Autonomous Systems (ASs) and Internet Service Providers (ISPs) [13]. Since P2P increases the inter-domain traffic and, therefore, creates large costs for ISPs, they might place restrictions on the traffic to limit the impact on their costs. Thereby, the transmitted amount of data would be forcefully reduced, packets would be dropped, and perceived video quality would degrade.

An example of application layer policies that affect resilience is the buffering policy. If pieces, which will soon be needed for play-out, are received too late, there will be stallings or disruptions in the video play-out. The pieces would have to be retransmitted, which would lead to higher bandwidth usage and increased delay.

3.2.4 User Layer

Another problem for resilience but at the user layer is how to counter free riding. Free riders aim at utilizing system resources without actually contributing to other peers. When designing solutions to hinder free riding one has to keep in mind, that there are peers, which have only limited upload bandwidth and, therefore, cannot contribute to a large extent. Also malicious peers pose a problem. They are likely to send no or even corrupted pieces.

It is worth mentioning that to effectively address a certain challenge at one layer, a tradeoff has to be made in order to calibrate the system and avoid performance degradation at a different layer. For example, to counter piece loss, a system designer can increase the amount of redundancy added to a stream allowing more resilience against packet loss. However, this comes at the cost of higher bandwidth requirements. Therefore, a tradeoff between the amount of required redundancy and bandwidth requirements has to be made.

3.3 Resilience Mechanisms

The literature on how to introduce resilience to P2P video streaming is vast. Within this paper we classify these mechanisms either as *core* or *cross-layer* mechanisms. A classification of the different mechanisms is presented Figure 2.

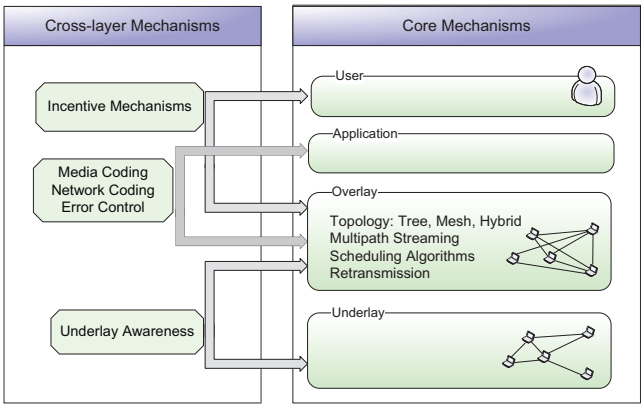

Fig. 2 Classification of resilience mechanisms grouped as core and cross-layer mechanisms.

3.3.1 Core Mechanisms

Core mechanisms are building blocks that have to be included in the system. Therefore, they constitute the design decisions a system designer has to make. For example, there exist various streaming *topologies* that can be used each affecting resilience differently. Possible topologies include tree, mesh, and hybrid. Additional core mechanisms include: *multi-path streaming*, *scheduling algorithms*, and *retransmission*. Since in this paper we focus on P2P video streaming, only core mechanisms at the overlay layer are of interest. Core mechanisms are presented and analyzed in Section 4.

3.3.2 Cross-layer Mechanisms

Cross-layer mechanisms can be applied to further enhance resilience. Cross-layer in this context means that different algorithms with information available at different communication layers would interact to bring more benefits and resilience to the streaming system. First, *incentive mechanisms* are cross-layer approaches between the user and overlay layers. Second, *underlay awareness* is a cross-layer approach between the underlay and overlay. Interaction between the overlay and application layers gives the possibility to add various resilience mechanisms, such as: *media coding*, *network coding* and *error control*. Cross-layer mechanisms are presented and analyzed in Section 5.

4 Core Mechanisms

Core mechanisms constitute core functionalities that have to be there in any P2P streaming system. The most influential core mechanisms and which will be the focus of this section are topologies, multi-path streaming, scheduling algorithms, and retransmission.

Topologies are the focus of research when it comes to P2P streaming. Two basic delivery topologies have been proposed: tree and mesh. For both, adaptive and robust streaming strategies are necessary to handle dynamics and unreliabilities in the system. After we present those basic topologies, we present how combining both of them into one hybrid topology can bring more resilience to the streaming system.

4.1 Tree Topologies

In tree topologies, peers can be arranged as a single tree or in multi-trees. Video packets are then disseminated to all peers by making every peer actively push video packet to its child peers [3].

While single tree topologies have advantages such as easy implementation, there exist several shortcomings as described in the following subsection. Ghoshal *et. al* state in [16] that there is a need for additional algorithms to overcome the effects of churn in tree-based systems. Therefore, multi-tree topologies have been proposed.

4.1.1 Single Trees

Users participate in a video streaming session and build a tree at the application layer with the video source server being the root [3], as presented in Figure 3. When peers join, they are inserted at a specific tree level and their parent peers forward the content to them. Examples for the use of single tree streaming are Overcast [17] and ESM [18].

Tree Depth and Fan-out. Tree depth is defined as the number of levels the tree has. Therefore, the smaller the tree depth, the lower is the delay for the leaf peers (those situated at the bottom of the tree). To achieve a small tree depth, peers need to have a large fan out degree, which specifies, how many child peers each peer has. Since a peer has only a limited upload capacity, its maximal fan out degree is constrained. To provide load balancing and failure resilience, the fan out degree is normally set less than its maximum possible value.

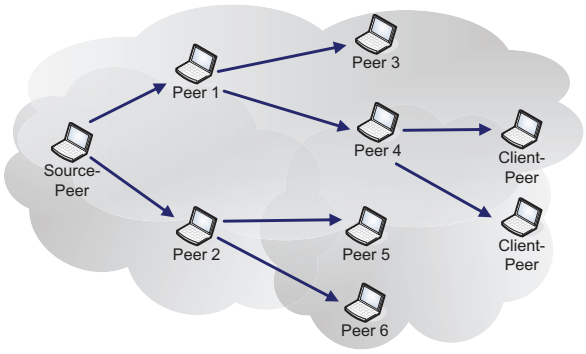

Fig. 3 A single tree topology.

Peer Departure Problem. Single tree approaches are strongly affected by ungraceful peer departures since after the departure of a peer all of its child peers lose their connection to the video source and, therefore, their media quality suffers significantly. The higher in the tree a failing peer is located, the expected the degree of disruption can be. Although specific recovery algorithms exist, recovery is still slow in the presence of frequent churn [3].

Upload Bandwidth Issues. One issue in tree topologies, is that leaf nodes do not utilize their upload bandwidth as they do not have any children. This leads to reduced efficiency, as the number of leaf peers can be rather high [19]. In [16], it is explained that system throughput is limited by the link with the lowest capacity among all links on the path from the source to the leaf peers, which limits system efficiency. Another major issue of tree topologies comes from the assumption that peer upload bandwidth is double the video bit-rate, since a peer must typically upload twice as much as it downloads. This is, however, contradicting with average link capacities, where upload capacities are usually much smaller than download ones, rendering tree topologies un-usable in real life scenarios. In the next subsection, we discuss how these issues can be overcome by using multi-tree approaches.

Churn Problem. An issue that still remains a research topic is how to ensure robustness and resilience in tree-based topologies with high churn rates and in the presence of flash crowds, where many peers join or leave the network simultaneously. In this case new peers need to be integrated quickly into the structure without reducing video quality of all participating peers, respectively the delivery structure has to be repaired to minimize service disruptions [14]. Tree topologies are best suited for networks with restricted available bandwidth and stable peer memberships, since their control overhead is lower, as in mesh approaches [16].

4.1.2 Multi-tree Streaming

In multi-tree approaches, a stream is divided into several sub-streams at the server. For each sub-stream, a subtree is created. A peer must join all subtrees and receive all sub-streams in order to decode the video stream. The power of this approach comes when each peer joins each subtree at a different position.

Figure 4 shows an example of how peers can be positioned within the different subtrees. If a peer is in an

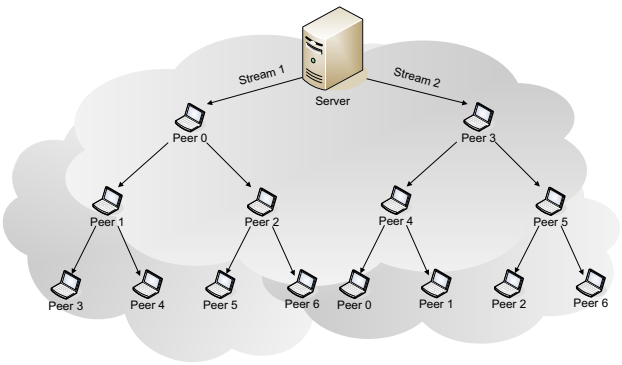

Fig. 4 Peers participating in a multi-tree topology. Here, each peer joins two subtrees.

inner node in a single tree, all of its upload bandwidth is thoroughly utilized and, therefore, the frequency of being placed as an inner node in a multi-tree topology should be proportional to the available bandwidth [3]. Although having multiple subtrees and the possibility to send packets in different sub-streams over different paths, delivery paths usually remain static just like in single tree approaches. In addition, the underlying physical topology cannot be neglected for efficient content distribution and there exists a trade-off between minimizing the tree depth and the network path diversity [4].

Multi-tree Construction. Construction can either be done in a randomized or a deterministic manner. Using a randomized construction, a search for nodes with spare bandwidth is started in each tree. Then one of the hit peers is randomly chosen as parent peer for a child peer. For deterministic construction each node is assigned to be interior node in only exactly one tree to shorten the tree. Disjointed interior nodes in each tree lead to increased tree diversity and, therefore, robustness. According to the deterministic algorithm, the first decision that has to be made upon a new node arrived, in which tree it should be fertile. Thereby, fertile refers to it becoming an interior node, which could have child peers. In all other trees this node is so-called sterile. The algorithm aims at balancing the number of fertile nodes in all the trees. Parent nodes are chosen deterministically for insertion of a node and randomization becomes unnecessary due to disjointness of interior nodes.

Examples. Some examples of multi-tree approaches are SplitStream [20] and CoopNet [21]. CoopNet aims at reducing the dependency of a peer on its parents. Typically, video coding techniques are also combined with multi-tree streaming, for example CoopNet utilizes Multiple Description Coding (MDC) (described in Section 5.3.2). There, the departure of a node in a multicast tree does not completely deprive its child peers, because they can still gather most of the sub-streams from other trees that they are part of. Resilience is furthermore increased because a peer can only be an internal node in one tree, and, therefore, its departure disrupts only the connections and sub-streams in this single tree. However, this comes at the cost of a more complicated setup and maintenance algorithms.

For the construction and maintenance of the trees, a centralized tree management algorithm for multi-tree topologies is introduced in [22]. The main methods of this algorithm to ensure resilience are: short trees to minimize the probability of disruption and making a trade-off between tree diversity and efficiency. According to this algorithm, the root peer coordinates all tree management functions. It handles join-events by feeding new nodes with information about their parent peers in each tree. Upon being informed of a graceful node departure, the root node assigns new parent peers to the children of the departed node in each tree. Ungraceful departures are handled by measuring the packet loss rate, which is measured in each tree by each peer. If the rate exceeds a certain threshold, a peer checks if its parent in the same tree also experiences this problem and if necessary sends a request for a new parent to the root peer. Requests are stored and could be used for future parent selection.

As a summary, multi-trees are more resilient compared to single trees, and the upload link capacity of peers is better utilized [14]. In addition, they give rise to more interesting cross-layer techniques, such as media coding and Forward Error Correction (FEC).

4.1.3 Redundant Trees

In [23] a protection approach called "redundant trees" is introduced. Since it is a protection mechanism, it is pro-active and is - in contrast to a restoration mechanism - not applied after a failure. The algorithm builds two directed multi-trees with disjointed links to provide resilience to failures of both peers or links. One of the trees is the main routing tree and the other tree is created as a backup in case of a structural failure on the path of the main tree. After a failure, successful transmission can be accomplished by path rerouting over an alternative path in the backup tree. Back-up routes are preplanned and, therefore, the algorithm provides faster recovery than dynamic schemes, in which path computation is done only when a failure occurs. To cope with multiple failures, more than one backup tree could be created. However, there is a limit on how many back-up

trees can be created and managed that depends on the specific scenario and peer behavior.

4.2 Mesh Topologies

Mesh topologies, such as that shown in Figure 5, enable massive parallel content distribution among peers. They are based on self-organization of nodes in a directed mesh and do not have a static topology [3]. Connections are based on availability of content and bandwidth. While peers in a mesh have more links and thus better connectivity, the mesh content delivery scheme, which is known as swarming, is more complicated [16]. To allow for mesh streaming, a video file is subdivided into many small pieces typically ranging from 32kB to 512 kB, however, piece sizes of several megabytes have also been used.

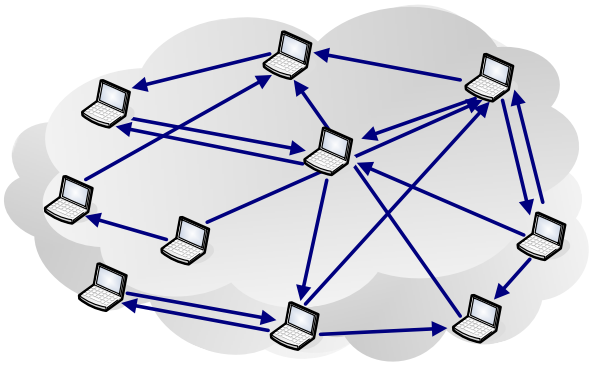

Fig. 5 An example mesh topology.

Every peer would request the pieces about to be played out from other peers in its neighborhood. To find out which peer has which piece, so-called buffer maps (bitmaps of available blocks) are periodically exchanged between the peers in the same neighborhood.

Structure. Mesh topologies can be unstructured or structured. In unstructured meshes, peers are connected to randomly chosen peers to provide more neighbors and different delivery paths. This leads to robustness when failures occur, since every piece can be obtained from other peers in a simple way [14]. Examples of unstructured meshes are PRIME [24] and CoolStreaming [1]. In structured meshes, on the other hand, peers are typically arranged into clusters, with the majority of links being established within the same cluster. The peers, which connect the different clusters can be considered as super peers and should have good bandwidth and availability characteristics [16].

Advantages. The advantages of mesh-based approaches are low costs and simple maintenance of the network structure. Compared to tree-based systems, these topologies are more resilient regarding the topology in the presence of node failures and departures, as the probability of more paths being available is higher [4]. In [25], Magharei *et. al* discuss that paths from the source to individual peers are more stable than in tree based schemes and that a peer's degree of stability increases the longer it remains in the overlay.

Overhead. The main disadvantage of mesh-based topologies is the overhead that comes from the periodical exchange of buffer-maps and status messages among the peers. These messages can become rather large especially with long or high definition streams. Decreasing the number of pieces relaxes this overhead, however, at the cost of more dependence on peers that a certain pieces has been assigned to. Since the piece then can become rather large, fast peers might get stuck with slow uploaders.

Comparison to Tree Topologies. When compared to tree-based approaches, streaming over mesh topologies is more resilient against peer dynamics, but it also has the disadvantage of extra overhead due to the non-existent parent-child relationship for routing that requires the exchange of buffer-maps. This leads to higher delay and more control overhead. Furthermore, overhead grows if control notifications are sent for every received piece. This issue can be countered by the use of larger pieces, since overhead is reduced. Unfortunately this also comes with the cost of increased latencies. The trade-off between efficiency and latency is a main problem for mesh-based approaches, and, therefore, for some application scenarios, it can be better to use multi-tree streaming [16]. Nonetheless, for networks with high bandwidth and high churn rates the use of mesh-based streaming is far more appropriate than using tree-based streaming, due to its higher resilience [16].

4.3 Hybrid Topologies

Mesh topologies have the advantage of simplicity and resilience, but they suffer from higher latency and communication overhead. Tree topologies have inherent instability, maintenance overhead, and bandwidth utilization issues. Therefore, researchers have proposed the combination of both tree and mesh approaches in a hybrid overlay incorporating the advantages of both topologies while trying to avoid their disadvantages. Such system include: Chunky-Spread [26], mTreebone [27], and CliqueStream [28].

Figure 6 shows how a hybrid streaming topologies could look like. Next, we give more details about the Chunky-Spread and mTreebone systems.

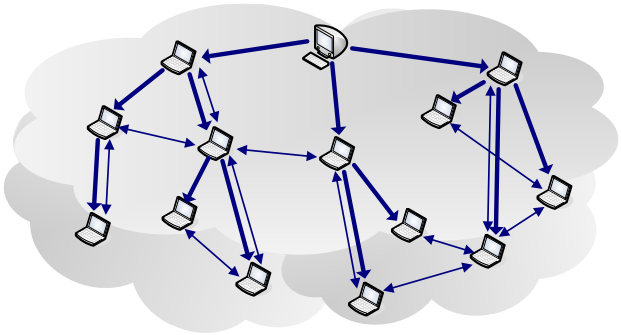

Fig. 6 A hybrid topology consisting of tree and mesh topologies.

Case Study 1: ChunkySpread. In this system, the stream is subdivided into several slices, which are distributed over separate trees, furthermore the nodes build a neighborhood graph [14]. Tree construction and maintenance are simplified due to the hybrid design and because it handles changes in memberships efficiently, the network overhead can be reduced [16]. As described in [26], the number of neighbors a peer has, depends on the peer's load. The higher its load, the more neighbors are assigned to it. Each peer knows the minimum target load of all other peers and it is also aware of the node degree associated with that minimum. In presence of churn, the actual neighborhood set changes, to ensure that the amount of neighbors for a node remains approximately the same over time. Local information about load and latency are exchanged periodically among neighboring peers and can be used for determining the appropriate parent-child relationships for each of the trees. To ensure even load distribution, each node periodically checks for overloaded parent peers and underloaded peers, which potentially could take up the role of the overloaded parents. If necessary, these peers then change their roles.

Case Study 2: mTreebone. In this system, a tree-based network is initially created. It consists of the stable network nodes and provides the streaming backbone. The unstable nodes and leaf nodes are arranged into a mesh structure, which is built on top of the backbone. The system uses a push/pull switching mechanism for content delivery and there is the need for a high degree of coordination between the backbone and the mesh structure. This hybrid overlay provides some additional resilience against churn, however, it still depends on locating alternative stable nodes after one leaves the backbone.

The advantage of this approach is that a small tree-based backbone can lead to better efficiency and low latency. However, the initial time taken to build the backbone can vary. Contrary to other topologies, robustness and resilience can be achieved even when facing high churn rates and many disconnections of peers located in the backbone. In this case the content dissemination is done by the mesh overlay until the backbone is reconstructed [16]. Although only a single tree backbone is created, a multi-tree backbone would also be possible [14]. The key challenge lies in the identification of stable nodes and their positioning at appropriate tree positions.

4.4 Multi-path Streaming

As discussed above, an efficient streaming overlay requires the establishment of multiple paths between the different peers sharing a media stream. In this subsection, we present how the effects of transmission errors can be effectively overcome by path diversity by using logically as well as physically disjointed paths between different peers. As an example, we show in Figure 7 how video packets can be sent over multiple different paths within a multi-path streaming overlay.

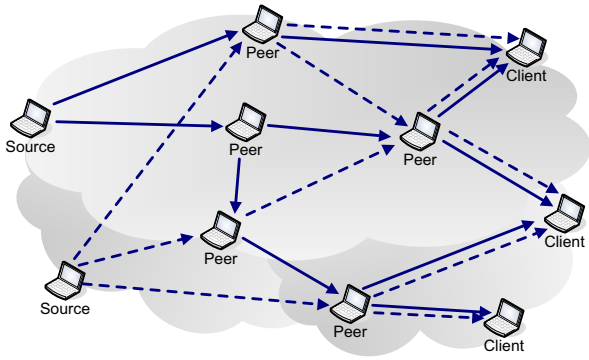

Fig. 7 Video packets can be sent over multiple different paths.

Challenges. The multi-path design introduces various challenges namely how to choose the paths efficiently and how to manage rate allocation to have an efficient approach that can adapt to dynamic streaming architectures. Abdouni *et. al* [29] state that in addition to increased bandwidth utilization, a better adaptation to changing network conditions is possible. They focus on loss characteristics and optimal load distribution among the multiple paths under the consideration of different path characteristics. Furthermore, they conclude that the performance of single path streaming can be increased by using forward error correction techniques.

Advantages. One of the advantages of this design is the potential decrease in delay. In addition, disjoint transmission paths with relatively independent losses allow for better error resilience. The power of multiple

path streaming lies in the usage of the multi-path nature of P2P networks to meet bandwidth requirements by aggregation of network resources. It is worth noting that paths do not have to be completely disjointed, as it is sufficient if they have disjoint points around congestion or bottlenecks [30].

Spatial versus Temporal Diversity. Rather than using multi-path streaming and, therefore, achieving spatial diversity by sending pieces over different paths, one could try to achieve temporal diversity by sending pieces over the same path at different points in time. But since this approach suffers from additional delay, it would not be applicable for applications with strict real-time constraints [29,31].

Mechanism Optimization. Multi-path techniques require the use of additional methods in order to make it more powerful. For example, in [32], Karrer *et. al* state that there is a need to combine multi-path streaming with other mechanisms to adapt, for example, fluctuations of resource availability. Karrer *et. al* introduce the two concepts of multi-path streaming at both the application layer and the transport layer. At the application layer, data transport is integrated into the application context and, therefore, enables the combination of adaptation and multi-path streaming. To ensure that both the same data is sent over different sub-streams and the most important frames are sent first, multi-path streaming has to be combined with filtering. Its performance is decreased if the number of paths is high, thus the transport-layer approach is more suitable if the number of parallel streams is high.

Further work on multi-path streaming includes [33], which proposes a selection scheme for two node-disjoint paths. In [34], Apostolopoulos *et. al* describe source and path selection mechanisms based on disjointness and jointness of network segments. In [35] it is explored how Forward Error Correction (FEC) can be utilized for multi-path streaming and what effects this has, while [36] gives a state of the art on existing multi-path streaming approaches.

4.5 Scheduling Algorithms

Scheduling algorithms are used to determine which pieces should be (re-)transmitted and at what time these transmissions should be carried out [37]. The scheduling decisions also include dropping of pieces if their transmission is not possible due to restricted resources. The aim is to maximize the quality of the video stream, without causing a waste of network resources. If a suitable scheduling algorithm is deployed, it is not only possible to reduce end-to-end delay and achieve more robustness and adaptive resilience against errors [38], but also

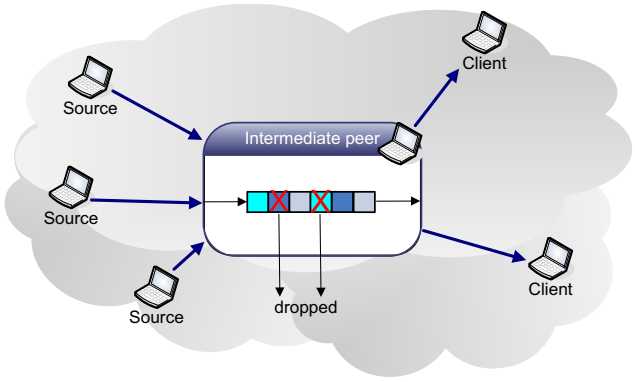

Fig. 8 Based on distortion information, (intermediate) peers can make rate-distortion optimized scheduling decisions for incoming pieces.

bandwidth distortion between peers across all channels can be minimized [39]. Moreover, in case peers drop some pieces for assurance of timely delivery of more significant pieces, video quality can be supported by optimized scheduling.

To achieve optimized scheduling, prioritized transmissions, receiver feedback and retransmissions can be used. Optimization can be based on both distortion of congestion or rate. Figure 8 shows an example scheduling decision based on rate-distortion information. Structure and transmission characteristics (for example available bandwidth at the different links) of the network should also be considered. During scheduling, not only the packet transmission order has to be determined, but also a prioritization among a peer's descendants should be taken into consideration, since they would be affected in the case of piece loss or delayed arrival of the pieces [37]. Furthermore, successive transmissions can be spaced to achieve congestion avoidance at bottleneck links as well as to limit delay of control packets. Not only determination of the order in which packets have to be sent is required, but also prioritization of the neighboring peers.

Scheduling algorithms are crucial for achieving a good quality of experience. One reason for this stems from the strict timing requirements of streaming applications. Furthermore, peers are heterogeneous, which means the available resources vary among peers. One peer in a network might have a high amount of available bandwidth and, therefore, is able to receive all video pieces in time, leading to high media quality. Simultaneously another peer in the same network can have only small amount of available bandwidth and thus the scheduler has to determine which pieces are the most significant to ensure that the video can be watched at a certain quality level. As stated in [37], approaches based on rate-distortion optimization can have high

complexity and, therefore, congestion-based optimization might be better. If data is distributed by several senders the question arises as to how scheduling can be deployed. There, an additional objective is to avoid any waste of resources due to the transmission of redundant pieces. To achieve this, sending peers must coordinate their scheduling strategies [40]. As scheduling imposes an additional real-time computation burden, an initial off-line processing time might be useful. While this can be done easily in VoD systems, it cannot be applied to live streaming systems, because video streams are not pre-stored.

4.6 Retransmission

Research on retransmission strategies for packet-based systems is well known in the client/server communication community. However, such strategies have not been investigated enough in the P2P community. For P2P-based video streaming it is sometimes not always optimal to rely on lower transport layers (e.g. TCP) to handle retransmission of lost packets. In general, more sophisticated algorithms in accordance to overlay requirements and state could be developed.

In general, in a retransmission-based system, lost packets (which were not acknowledged on time) are simply retransmitted. There are different possible retransmission schemes such as Automatic Repeat reQuest (ARQ), parity retransmission and delay-constrained retransmission.

ARQ is a form of backward error correction and is usually used in unicast protocols [41]. ARQ of urgent packets is helpful only if it is possible for these packets to arrive on time if retransmitted. Retransmission would be done either after timeout or on explicit requests by a receiver. ARQ-based techniques are especially useful when path statistics are unknown [37]. While the utilization of ARQ is beneficial for the recovery of few errors or when the network undergoes temporary congestion, there exist situations where too much redundancy would be necessary for direct recovery. In this cases, additional FEC techniques should be applied.

Since it is possible to request a missing packet from another peer and not only from its original sender, ARQ in a P2P network is more complex. Therefore, the workload of peers and path characteristics should be considered while selecting which peer should retransmit a certain packet. In [41] it is argued that ARQ can lead to reduced throughput, since in multicast packet loss is often correlated with local throughput drop. In Figure 9, we present a possible P2P streaming system with ARQ-based retransmission.

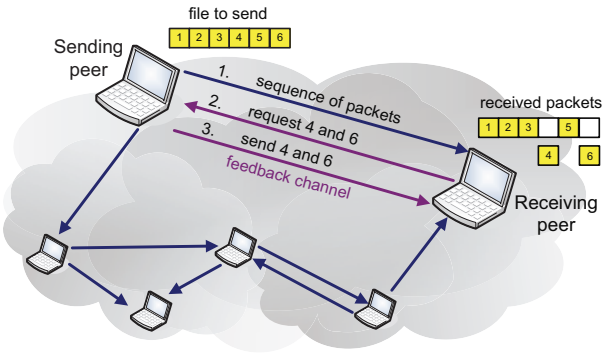

Fig. 9 In ARQ-based systems, a retransmission request is sent for each lost packet.

ARQ schemes have one important requirement, which is the availability of a reliable feedback channel. In addition, acknowledgment overhead is a drawback for ARQ in large networks. Sometimes, and in larger groups, the chance of uncorrelated packet loss increases, leading to the so-called feedback implosion problem. This happens because the sender does not have comprehensive knowledge about which packets have to be retransmitted to which receivers. Retransmissions that do not arrive on time lead to duplicate transmission of the same packet. In such cases, the system is better off using FEC schemes to enable a receiver to recover from errors without any additional network intervention. It is worth noting that retransmission of packets is helpful only when one way trip time is short enough with respect to the maximum permitted delay.

Usually, retransmission schemes are combined with FEC techniques. In such systems, it is possible to retransmit only a limited set of missing packets to be able to decode the video stream. This is known as selective retransmission. The major research question here is how to balance the amount of initial redundancy introduced into the stream with the expected amount of retransmissions. This aspect strictly depends on the media buffer size and the round trip time between peers [42]. For more information about retransmission strategies in P2P systems, please refer to [43, 37, 44, 45].

5 Cross-layer Mechanisms

In this section, we present and discuss cross-layer mechanisms that can be deployed in addition to the core mechanisms in order to further enhance resilience. These mechanisms can be classified as incentive mechanisms, underlay awareness, media and network coding, and error control.

5.1 Incentives

System performance can suffer significantly if peers do not contribute their resources either by reducing upload bandwidth or leaving the system all together once the video has been delivered. Therefore, incentive mechanisms are crucial to incite users to contribute their resources, achieving a more robust and performant system [46]. These mechanisms can be based on payment, punishment or service differentiation, as explained in [47]. Its worth mentioning that many deployed closed-source systems do not implement incentive mechanisms as the user does not have a choice in reducing his or her contribution. Nonetheless, incentives are being considered for such systems [48], since users might still throttle the traffic of the streaming application. Other systems [49], count on the social relations of the users and their explicit consent to allocate resources to specific users.

Incentive mechanisms for classical P2P file sharing applications are well established, e.g. tit-for-tat that is used in the BitTorrent system [50]. However, such mechanisms are not suitable for streaming applications mainly due to the real-time requirements of these systems. Since a piece close to the playback position has to be quickly received, there is no time to establish efficient bartering relations. This problem is evident especially when peers have different playback positions, which is normally the case for video-on-demand. An additional challenge is the skewed resource contribution of peers. As stated in [48], the majority of system capacity is often contributed by a minority of high capacity peers. Therefore, it is quite crucial to have incentives for these peers since without them, the system would collapse. These peers can be rewarded through better Quality of Service (QoS) in terms of robustness [48], or through offering higher video quality, for example when using layered video coding [51].

Incentives can be broadly categorized as either bilateral or multilateral. Bilateral incentives are based on local observations [52], while multilateral incentives are based on reputations delivered by other peers or even managed globally [48].

Multilateral incentives are generally more applicable to streaming systems, as they better cope with real time streaming requirements. Multilateral incentive mechanisms usually rely on so-called *reputation systems* to assess peer cooperation over a longer period of time. In the next subsection we give an overview on some prominent incentive mechanisms for streaming systems. After that, we present an overview on reputation systems.

5.1.1 Prominent Incentive Systems

Habib *et. al.* present a score-based incentive mechanism in [47]. In this system, rational users choose their contribution level according to an own utility. The contribution level x_i of peer i is mapped into a score S_i, which is used to determine a percentile rank R_i. This rank designates the relative ranking of the peer compared to all other peers in the system. A higher rank can be achieved by higher contributions and allows for selection of better serving peers (resulting in better streaming quality). A peer is allowed to select serving peers with the same or a less rank. The peer utility U_i can be expressed as a function of the streaming session quality Q and the contribution cost C, therefore:

$$U_i(x_i) = a_i Qi - b_i C(x_i),$$

where a_i and b_i weight streaming quality and contribution cost for user i respectively. Typically, a peer will determine its optimal contribution level to achieve maximized utility. Newcomers to the system are assigned a score of zero; hence, they receive best-effort service. To prevent free-riders exploiting other users, newcomers and free-riders are treated the same. Additionally, to encourage continuous contributions, the scores can contain an aging factor.

Another approach to incorporate incentives is presented in [48] by Piatek *et. al.* where the authors integrate their solution, called *Contracts*, into the PPLive live streaming system. They argue that in live streaming systems, cooperative peers cannot be rewarded by higher download rates (since the content is generated while being streamed). The authors conclude that bilateral incentive strategies are not sufficient for live streaming. Therefore, they reward globally effective contributions with increased robustness in the presence of churn. Agreements concerning the contribution evaluation are formed between clients and the system. The developed system also accounts for heterogeneity of peers by structuring the topology accordingly; peers can reorganize their position in relation to the streaming source. Thereby, peers with higher capacity and higher contributions are placed closer to the source and receive faster service. By additionally taking into account the underlying network topology, a more consistent performance increase can be achieved. For topology updates, an incentive-based gossip protocol is utilized. To prevent malicious behavior, receipts are verified in both a centralized and distributed fashion.

5.1.2 Reputation Systems

Reputation systems aim at keeping a history and estimating how well peers cooperate and contribute to

the system. Reputation systems are required for multilateral incentives to work, since a history of peer cooperation is needed. We now give an overview about reputation systems. A more elaborate analysis is given in [53].

In general, reputation can either be managed globally or by combining different local experiences. A focus on either positive or negative experiences is possible, as well as a combination of both [46]. In some systems [54, 55], a peer has the possibility of enhancing reputation reports it has received from other peers. For example, it can include perceived transaction time. Other systems [56, 55] require the existence of so-called mediator peers. These mediators help in reducing the complexity of reputation systems by having more trustworthy entities. However, it requires additional information to achieve transparency of mediation. This design can be further extended to have a centralized and trustworthy entity that collects and aggregates all reports.

But a main question in many systems remains: how can one determine whether a certain peer is trustworthy or not? In [57] an approach is introduced, in which trustworthiness of peers is determined by direct observations and through polling of witnesses.

In any case, reputation systems still suffer from various issues. For example, through collusion a peer can achieve a higher reputation by agreeing with another malicious peer to falsely send positive reports about each other. A peer can also receive a wrongly negative rating from a peer, that wants to increase its own reputation in relation to the defamed peer. Furthermore, peers can provide random information about other participants, so that they achieve better reputation although they do not contribute to the system. It should also be noted that peers can change their behavior over time, therefore, it is important to distinguish between outdated and fresh reputation information. Additionally, a main issue in reputation systems remains the management overhead for storing, maintaining and querying reputation information [58, 57]. Therefore, a system designer has to be careful that an incentive mechanism does not introduce overhead into the system that can outweigh the benefits.

5.2 Underlay Awareness

Transmission at an overlay network might be less efficient than transmission at the network layer, especially when the overlay network is randomly constructed. This stems from the fact that the distance[5] between peers, which are close in the overlay, can be large in the underlying network [59]. This not only leads to additional routing overhead and inefficiency, but also to an increased failure probability due to transmission over long paths with a possibly higher number of unreliable peers. To overcome these problems and provide better QoS guarantees, overlay construction and management could be done while considering the underlying network.

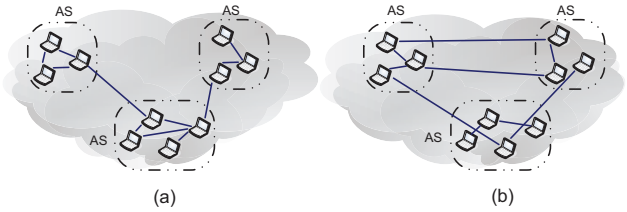

Fig. 10 (a) An ISP-aware topology based on biased neighbor selection versus (b) a randomly connected overlay.

In [13] different approaches that can be used to achieve underlay awareness are investigated and classified according to collection and usage methods. Four different underlay parameters were identified, namely *ISP-location*, *latency*, *geo-location*, and *peer resources*.

In ISP-location awareness, location information about the Internet Service Providers (ISPs) through which a peer accesses the network, is utilized in order to achieve greater locality of traffic. This entails several benefits for ISPs including reduction of costs and congestions. Furthermore, it leads to an enhanced quality of experience for users, since an ISP can make better provisioning of its network for its own locality-aware customers.

By considering latency to build a P2P system, better estimation and management of streaming delays are possible. In [13], it is explained how latency information can be measured and estimated. Further, the use of geolocation to construct an overlay leads to the introduction of innovative location-based services. This is particularly important for next generation IPTV systems that aim at integrating video streaming with location-based social interaction.

The last underlay information analyzed focuses on peer resources. Peer resources include available bandwidth, storage space, up-times and processing power. Collecting and using information about peer resources is important for achieving a good offer/demand balance.

Now we present some examples of underlay-aware P2P systems. Starting with mOverlay [59], which is a system in which the locality of peers in the underlying network is exploited by the utilization of the group concept. A group includes a set of nearby peers and neighboring peers in a group that acts as dynamic landmarks, with the aim to increase scalability and robust-

[5] Distance in this context can indicate geographical and well as topological distance.

ness when compared to approaches based on static landmarks. Another system utilizes the concept of the proximity neighbor selection as presented in [60]. There, routing table entries are selected in a way so as to ensure that the table contains the nearest of all peers in the topology. Figure 10 presents how the concept of biased neighbor selection can help in reducing inter-AS traffic by limiting the number of overlay links across different ASs. In Leopard [61], which is a geo-location awareness approach, the system tries to achieve constant routing stretch and load balancing during flash crowds. In DagStream [62] a peer selects geographically close peers as neighbors and can, therefore, minimize the packet delay between itself and its parent peers. In ZIGZAG [63] and AnySee [64] a peer chooses neighbors in a way that ensures minimal total delay between itself and the source.

With utilization of underlay awareness, the user can experience a better quality of experience due to the better quality of server peers, for example due to lower delays and shorter download times. The greatest contributions to resilience are introduced by having peer selection and overlay management consider underlay information that mostly affect resilience.

5.3 Media Coding

In this section, media coding an important building block of resilient P2P video streaming, is presented and analyzed. We generally consider media coding schemes that enhance resilience by allowing the receiver to better react to losses and bandwidth fluctuations. We have established a classification of the different media coding techniques. This classification is presented in Figure 11.

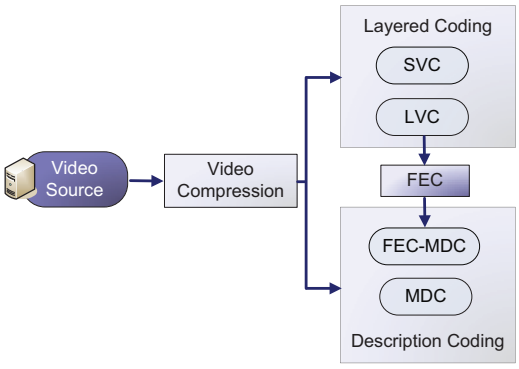

Fig. 11 Classification of different possibilities a video encoder can use to generate different types of video structures. These can be generally classified as either layered or description coding.

Raw video data is usually too large to be sent over normal networks, therefore, it is necessary to compress it, exploiting the high redundancy of video frames. For this purpose, video compression can be used to generate either layered or description-based video streams.

Layered Video Coding (LVC), on the one hand, is a term used to describe a large set of coding schemes that organize video data into layers. In LVC, a video stream is structured into hierarchical layers, where the so-called base layer represents the minimal amount of data in order to decode the video stream. To decode a certain layer, all layers below it have to be available. A prominent example of layered video coding is the Scalable Video Coding (SVC) standard [65]. SVC is presented in Section 5.3.1.

Description coding, on the other hand, is based on encoding video data into descriptions that do not depend on each other. The more descriptions are received, the better is the video quality. Most prominent examples of description coding are Multiple Descriptions Coding (MDC) and description coding based on Forward Error Correction (FEC-MDC) [37]. FEC-MDC is generated by FEC-encoding layered video streams. MDC and FEC-MDC are presented in Section 5.3.2.

5.3.1 Scalable Video Coding

Scalable Video Coding (SVC) [65] is the new amendment to H264/AVC: Advanced Video Coding standard. Based on the SVC design, a video sequence is separated into multiple layers. One of these layers is called the base layer, which provides coarse visual quality and can be decoded independently from other layers. All other layers are denoted as enhancement layers and can only be decoded together with the base layer. More available layers provide higher visual quality, but only with the complete bit stream is the highest possible quality achieved. This is the main drawback when compared to MDC, where it is not important which descriptions are received, as they can all be decoded independently.

An interesting application of SVC is the possibility of having intermediate peers with high capacity to dynamically extract layers from the scalable stream to serve peers, whose capacity is restricted. Further different encoding qualities within the same layered stream are possible, which enables efficient dynamic rate adaptation. Also the bandwidth of the network can be utilized better than in single layer coding. SVC is based on three modalities of scalability: spatial, temporal, and quality scalability, often called Signal-to-Noise Ratio (SNR) scalability. While temporal scalability enables different video frame rates for a stream, spatial scalability provides different video resolutions and SNR scala-

bility features different video quality levels. The coding efficiency is not as high as for single layer coding [66].

Quality Adaptation. SVC is utilized in various streaming systems, such as [6,67–69]. In [6] and [67], a mesh based video streaming system is presented, which utilizes SVC for support of adaption to available resources. Adaption is done based on different parameters: screen size and resolution, download throughput and processing capabilities. For adaption it has to be figured out, which resources are available and, therefore, what the highest possible streaming quality level is. Available resources are categorized: peer resources and P2P system resources. While peer resources include, for example, processing power and bandwidth, the latter resource type encompasses resources such as throughput and active neighbors. An SVC stream is at first separated into multiple pieces, of which each contains layers in the three dimensional quality space. During layer selection, which can either be performed with Initial Quality Adaption (IQA) or Progressive Quality Adaption (PQA) mechanisms, quality adaption is accomplished. An IQA mechanism is executed upon a peer starting to view the video. While this mechanism determines which is the most suitable quality level for this peer under consideration of its static resources, the PQA module aims at handling changes in network condition and available throughput.

In addition, [67] presents metrics to assess the performance of mesh-based P2P video-on-demand systems that uses SVC. These metrics are called *session quality* and *video quality*. Session quality metrics quantify how smooth playback is in terms of start-up delay and stallings, while video quality metrics quantify the overall video experience. In [67], it is shown that these two metrics exhibit a trade-off which has to be taken into account while designing and deploying such systems.

Cache and Relay. In [68], a streaming system is proposed, which combines cache-and-relay and layered-encoded streaming to deal with asynchronous user requests and heterogeneous peer bandwidths. The system incorporates a greedy approach, whereby the outgoing bandwidth of the peer, which has the smallest number of layers, is always maximally utilized. A peer, that wants to obtain the video stream, requests the individual layers from the set of supplying peers. In case the peer did not receive all expected layers and, therefore, did not achieve the desired video quality, it requests the missing layers from the server. In order to deal with the effects of node departures, a receiving peer has to carry out a layer allocation algorithm to reconfigure its session. During this time, the video quality may suffer. If a peer departures gracefully, it informs the receiver to reconfigure its session. The video quality is not affected if the reconfiguration is finished, before the sender runs out of data in its buffer. In the contrary case, the receiver can try to request the layers of the failed peer from the server, until the session reconfiguration is completed. This only works, if there is available server bandwidth. If not, the streaming quality of the receiver has to be adapted gracefully, which could also lead to reduced quality for its child peers. The authors concluded, that additional buffering helps to alleviate the propagation of quality degradation in certain circumstances.

5.3.2 Multiple Description Coding

Multiple Description Coding (MDC) [37] works by creating several independent so-called descriptions from a video stream. Each description is a video segment that contributes a certain amount to the video quality and can be decoded independently. The more descriptions are received, the higher is the received quality. As a simple example of how MDC works, consider the independent descriptions generated by dividing a video stream into even and odd frames. Alternatively, MDC can be realized by dividing video frames into independent sub-pictures by choosing odd and even horizontal and vertical lines of the picture, thus resulting in four descriptions.

MDC descriptions can then be distributed using mesh or tree topologies. MDC works well when combined with multi-path and multi-tree topologies since then it allows for better resilience against erroneous transmissions and playback distortions as presented in [37]. SplitStream [21] and Chunky-Spread [26] are two prominent approaches that utilize MDC to ensure resilience.

An interesting variant of classical MDC is the so-called *FEC-MDC* as presented in [37]. Similar to MDC, FEC-MDC has the same goal of dividing the video file into multiple independent descriptions but with the dis-

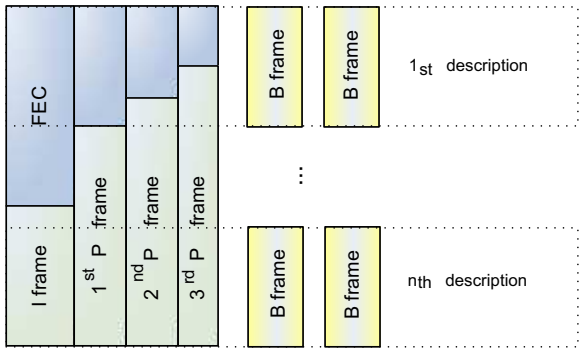

Fig. 12 By applying different protection levels to layered video packets, multiple descriptions are created and, therefore, constitute an FEC-MDC video stream.

tinction of using layered video coding and Forward Error Correction (FEC). The basic structure of an FEC-MDC coded video stream is depicted in Figure 12. The main advantages of this design in comparison to classical MDC is that it can use a classical video encoding scheme and offers more resilience through the use of FEC. As presented in [37], FEC-MDC starts by encoding the video data using any classical video codec to compress the video stream to generate a layered video streaming. Then, an unequal error protection is applied to the different layers in such a way that correction strength, achieved through more redundancy, depends on the importance of the layer. In comparison to MDC, FEC-MDC requires a certain minimum number of descriptions to be available before any useful video data can be decoded. This is due to the fact that FEC requires a minimum amount of available data so that it is able to decode the I-frames.

FEC-MDC has a better resilience against packet loss due to its built-in error control techniques and is flexible concerning the used video codec. However, FEC-MDC can inflict high amount of overhead. Nonetheless, this overhead can still be configured beforehand, through adjusting the level of redundancy, depending on system dynamics and loss ratios.

5.3.3 Layered versus MDC

In general, the main difference between SVC (or a general LVC) and MDC lies in the inter-dependency of SVC layers and MDC descriptions. Therefore, MDC clearly has the advantage of having flexibility and independence from a base layer and, therefore, simplifies mechanisms for the network building process. SVC approaches are the better choice for systems in which in-network scheduling is used for transmission, while MDC should be used for systems in which packet schedulers have no knowledge about both the importance of different packets and their interdependencies [70].

5.4 Network Coding

Another mechanism that can enhance resilience is network coding. Network coding is a technique where, instead of simply sending data blocks, peers in a streaming system can encode multiple blocks into one. The composite block would be sent and used to reconstruct the original data by decoding different encoded blocks from different peers. Its benefits have been analyzed in the context of multicast communication and P2P file content distribution, where it helped to increase information flow rates and decrease download time. Network coding helps in utilizing redundant network capacity, which would be otherwise left unused.

The basic procedure of network coding is described in the following. A stream is divided into pieces, which are again subdivided into n blocks $b = [b_1, ..., b_n]^T$. Each block has a fixed size of k bytes. To encode a new block x_j, a sender peer selects random, linearly independent coding coefficients for each of the buffered blocks. These coefficients $[c_{j1}, ..., c_{jn}]$ belong to the $GF(2^8)$ field. The sender then creates a coded block as linear combination of the buffered blocks: $x_j = \sum_{i=1}^{n} c_{ij} b_i$.

Other peers can re-encode received coded blocks and also send them to the receiver, which, upon receiving the first coded block, can start to decode the piece progressively using $b = C^{-1}x$, where C denotes the coefficient matrix, in which each row corresponds to the coefficients of one coded block. C is only invertible if its rows are linearly independent. Therefore, a peer has to receive enough linearly independent blocks in order to successfully decode the piece. For the layered approach proposed by [71], the chosen blocks for encoding have to belong to the same layer that is being requested, since some layers are not required by all peers in the network.

Recent research has shown how to utilize network coding for P2P-assisted live streaming [72] and how to deploy it in the context of a video on-demand system (*UUSee*) [73]. Moreover, *Chameleon* incorporates the combination of network coding and SVC [71]. Although this combination is not straightforward, since SVC tries to prioritize data of different quality levels while network coding aims at making all data pieces equally useful, it nonetheless helps to make layered P2P video streaming more performant and simplifies the streaming protocol. With *Tenor* [74], Shojania *et. al.* present a comprehensive toolkit, which provides efficient network coding methods and can be applied on a wide range of user devices.

Network coding enables the so-called "perfect coordination" where arbitrary peers can work together to serve the same piece for a receiver. As soon as the last coded block has been received, playback is possible. By using network coding, all coded blocks (from the different serving peers) become equally important (as long as they are linearly independent) and, thus, peers can collaboratively send blocks without high control and coordination overhead. This helps to significantly increase resilience. For example, if either a block is lost or a peer goes offline or is no longer uploading, this does not necessarily lead to decreased playback quality since other peers are able to send coded blocks for the required piece and can take over tasks of departed peers. Therefore, network coding has been posed as a solution to

the problems of rare blocks [73], long buffering delays, as well as high server bandwidth costs [71].

Challenges. Network coding also comes with some challenges and issues. One of them concerns the proper choice of block sizes. Smaller block sizes help to better utilize slower upload links, but lead to lower coding efficiency [73]. Another challenge stems from the exchange of piece availability information between senders and receivers, which depends on network coding factors. Whereas Liu *et al.* propose the utilization of a two-level cache map design [73], Nguyen *et al.* present an approach using two-bit buffer-map entries to indicate if a peer has received enough linearly independent blocks for decoding and/or serving a certain piece [71]. The major drawback of network coding, as presented in the research community, remains the high complexity, since the peers have to actively encode and decode blocks on the fly. Nonetheless, some recent works have shown that this complexity can be reduced using, e.g. Gauss-Jordan elimination and efficient implementations [71,74].

5.5 Error Control

To mitigate the effects of churn and congestion-induced losses on perceived media quality, *error control* through redundancy can be utilized in streaming systems. First, we describe *forward error correction*, then we present how resilience can be enhanced through replication and caching. Then we present *error concealment and recovery*, which can be used to minimize the effects of errors when they occur.

5.5.1 Forward Error Correction

When media packets are lost, perceived video quality at the receiver can suffer significantly. This loss occurs due to peer departure, network congestion and transmission over unreliable channels. To provide robustness and resilience in the face of packet loss and to overcome its effects, Forward Error Correction (FEC) can be applied to the video stream. In FEC, redundant information is added to ensure that the original information can be reconstructed when packet loss occurs. The striking advantage of this error correction scheme stems from the fact, that no further interaction between the sender and receiver is required for the sender to recover lost information. To ensure that the missing information can be reconstructed solely from the correctly received packets, the proper amount of redundancy has to be added. While the efficiency of recovery via retransmission drops significantly in systems with many receivers, due to most likely uncorrelated losses, this does not hold for FEC, since recovery does not depend on which packets are lost [41]. Figure 13 gives an example of how a video stream encoded with FEC can be decoded if some pieces are missing.

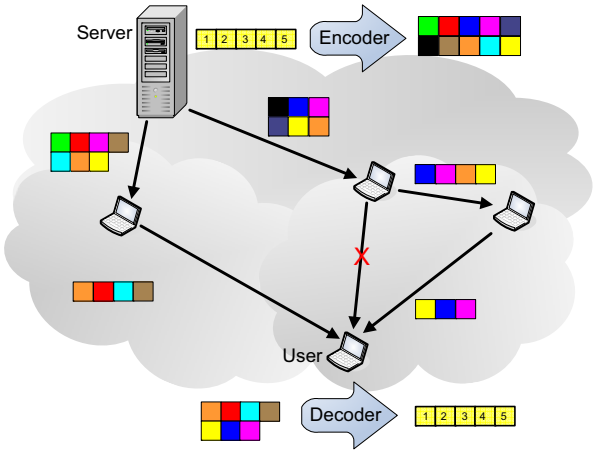

Fig. 13 Sending data with some redundancy, through forward error correction, allows receivers to recover from erroneous transmissions.

But one should keep in mind that the advantage of resilience comes with the cost of increased computational effort and overhead traffic. Optimizations such as unequal error protection and interleaving are possible.

Byte-level versus Packet-level FEC. When considering FEC codes one also has to differentiate between byte-level and packet-level FEC [75]. In byte-level coding, an encoded symbol (the basic unit that can be corrected) is in the order of bytes, whereas for packet-level coding it is as large as one whole packet. Packet-level FEC typically is required at the application layer, whereas byte-level schemes are implemented at the physical layer of nearly all wireless networks. Byte-level FEC is not able to restore a completely lost packet. This stems from the fact, that a corrupted packet is detected and discarded either at the link layer via cyclic redundancy checks or at the transport layer via checksum and, therefore, it is not available at the application layer. Since IPTV systems typically have to deal with the loss of whole packets, performance metrics for packet-level FEC are of interest for such systems. Nonetheless, byte-level FEC should still be used in a wireless streaming scenario, as then better error resilience can be performed by a specialized FEC, rather than relying on that of lower layers.

FEC for IP Channels: Erasure Codes. There is a difference between errors and erasures. Error symbols are erroneously received symbols that are undetected, whereas erasures are symbols which were surely received erroneously. Erasures are easier to correct as their position is known and typical FEC codes can better handle such errors. Usually, in an IP network, a

packet either arrives or not, therefore, packet-based erasure codes are particularly attractive.

In erasure codes, k redundant packets are created out of n original packets. A receiver only needs to receive k (error free) packets of the original n packets in order to achieve a perfect recovery of a certain block. Usually, the maximum number of missing packets that can be tolerated (accounting to $n - k$ packets) has to be calibrated according to system and network states so that redundancy is matched to the level of perturbations. Since missing packets can be the result of not only network congestions but also due to peer failure, erasure codes have potential in P2P scenarios.

There are a number of efficient implementations of erasure codes that can be used as presented in [76]. These include: Cleversafe[6], Luby[7], Zfec[8], and Cauchy Reed Solomon codes from the Jerasure library[9]. Experimental results indicate that the Cauchy Reed Solomon code shows the best performance.

FEC for Wireless Channels. In addition to handling erasures, FEC codes for wireless channels should additionally handle symbol errors. This is due to the fact that wireless channels tend to corrupt specific symbols. Although error correction capabilities are already built in lower layers, the design space of having a single FEC code for the all layers is attractive. Nonetheless, some research still needs to investigate whether this is cost effective and efficient. Prominent FEC schemes include Reed-Solomon codes (RS) [77], convolutional codes [78], and Low-Density Parity-Check (LDPC) codes [79].

Interleaved Reed-Solomon codes create different code words from normal RS codes by interleaving [80], thereby multiple code words are arranged into rows. Interleaved RS codes are mostly suitable for applications with burst errors, since they are most effective in case of correlated errors, which enable collaborative decoding.

FEC Code Optimizations. To enhance the performance of a streaming application, coding optimizations are possible [75]. These include: unequal error protection and interleaving

Unequal error protection is based on having knowledge about the different priorities of video stream packets. For more important packets better protection can be applied, while less important ones are not or less effectively protected. Typically, segments of a high priority stream should be provided with better protection. But since these streams have segments of various significance, it might not be overly beneficial to provide increased protection for all these segments. Instead, important segments could receive better protection.

Another possible optimization is interleaving that aims at increasing the error correction efficiency. While classical FEC is not able to safeguard packet transmission from extreme burst errors, FEC interleaving can mitigate such effects and lead to increased performance. FEC interleaving is often utilized for video streaming to mitigate the effects of correlated packet loss, for example due to congestion. FEC interleaving works by allowing the sender to re-order or interleave FEC-encoded packets before transmitting them. Therefore, originally adjacent packets are well separated from another by a distance, which can be configurable depending on the network state. Interleaving provides improved resilience against errors without requiring more bandwidth, which is especially effective when used for media streams with short-term dependencies between data packets. But this comes with the cost of increased latency due to additional buffering at the sender and, therefore, hinders the suitability of interleaving in delay-sensitive applications.

5.5.2 Replication and Caching

Content replication can be used to increase scalability and, therefore, resilience against network failures. Resilience is achieved by caching and replicating the same content, usually bringing the content closer to the users. Advantages include reduced bandwidth consumption and load on streaming servers. Another benefit is lower latency for clients and higher content availability as peers, when using some performance metrics such as latency, can locate a copy closer to them.

Examples. The manner in which data is allocated can be used to differentiate different replication schemes. In some systems such as Freenet [81], there are no explicitly chosen places for the replicated data. Replicas can be created implicitly by a peer when it requests the data. Deterministic replica allocation, as employed in applications such as [82], aims at alleviating discovery during requests. Furthermore, replicating peers can be chosen explicitly by predefined conditions as done in [83], to achieve enhanced resilience against correlated failures. To provide availability and consistency among replicas, reliable peers should be chosen. Gopalakrishnan *et. al*, state in [84] that a general P2P system should include caching or replication to provide reliability, low latencies and load balancing. They propose a minimalist replication scheme that only needs local information

[6] Cleversafe Inc., Cleversafe Dispersed Storage, open source code distribution: http://www.cleversafe.org/downloads

[7] M. Luby, Code for Cauchy Reed-Solomon coding, unencoded tar file: http://www.icsi.berkeley.edu/ luby/cauchy.tar.uu

[8] J. S. Plank, Jerasure: A library in C/C++ facilitating erasure coding for storage applications, Tech. Rep. CS-07-603

[9] Z. Wilcox-O'Hearn, open source code distribution: http://pypi.python.org/pypi/zfec

and is robust to dynamics. An example of how replication can be incorporated in P2P streaming systems is Overcast [85]. There, caching and replication strategies are utilized to increase robustness and resilience. According to [17], data is replicated at peers in such a way to ensure lower bandwidth requirements.

Advanced Techniques. By applying cache sharing and cache hierarchies a cache can access data, which is stored at another cache. This aids in distributing load more evenly among different peers and in overcoming the effects of network bottlenecks. Caching a whole video file (especially huge ones) would typically be inapplicable, since this would require large storage space. The investigations done in [86] state that peers can already experience increased performance, even if only important parts of the file were cached. Two video caching strategies are introduced in [87], namely: initial caching and selective caching. The latter can maximize the robustness of streams in the presence of network congestion without exceeding the limited size of the decoding buffer. Furthermore, using application and content hints was recommended in [88].

Caching versus FEC. In comparison to approaches that utilize low cost FEC, caching needs at least twice as much storage capacity. Situations where whole-file replication and erasure coding become beneficial are investigated in [89]. It is explained that replication can be more suitable than erasure coding when component availability is low compared to a certain threshold of storage overhead. This hold especially for P2P systems with low peer availability. When this threshold is exceeded, erasure coding with the maximal possible number of blocks should be used.

5.5.3 Error Concealment and Recovery

Random errors and short-term connection problems lead to reduced throughput that can cause quality degradation. In addition, relative long-term packet loss and network partitioning is possible. For the first case, recovery could be fast, but special error concealment and recovery mechanisms are needed for the latter case.

Typical FEC techniques are not sufficient for recovery from errors induced by long-term network partitioning due to continuous packet losses. In [90], Lui *et. al* propose a disruption-tolerant content-aware video streaming approach in which content summarization and error spreading techniques are combined to tackle problems arising from long-time disruptions. In error spreading techniques, errors are not corrected but evenly distributed among the remaining segments of the video sequence. In this approach, a set of so called summary frames is selected, which provide a visual summary of the video. While the understanding of the video content is not always possible if key frames are lost, this is not the case for summary frames, since they only provide an overview of the content. The visual content of summary frames can be replaced by the content of non-summary frames, as their play-out leads to a video sequence with near optimal quality regarding the original complete sequence. Summary frames are selected in such a way that there is a single complementary frame, which differs from the respective summary frame only to a small extent. Since the position of summary frames is ahead of their position in the video sequence, they can be played-out at the client in case of long-term disruptions. If summary frames are lost while a disruption occurs, video quality is not affected by their loss after the disruption has passed.

Another system that loosely fits within the category of this section is presented in [91]. This system aims at optimizing the streaming topology by synchronizing playback positions. Playback of different peers is synchronized by accelerating video playback (fast forward) of peers that lag behind. In addition to providing lower delays between different peers[10], such a system has the benefit of a topology that can be better optimized using more sophisticated construction algorithms.

6 Discussion and Conclusions

In this paper, we have presented a survey on various mechanisms for achieving and enhancing resilience in P2P video streaming. The different mechanisms were classified as either core or cross-layer. In this regard, the core mechanisms described design choices for the streaming system, while cross-layer mechanisms presented how additional mechanisms can be deployed to enhance resilience. Deployment and combination of these mechanisms for P2P video streaming faces several challenges, this will be the focus of our conclusions.

6.1 General Challenges

Building resilient P2P streaming entails several challenges, such as building an efficient overlay, which has desirable characteristics such as scalability and load balancing. Another key challenge is how to deal with peer dynamics, because constant churn leads to service interruptions if no countermeasures are taken. Therefore, robust and adaptive mechanisms, which manage

[10] This can be exceptionally important while streaming sport events, where people do not like to have delayed playback in comparison to neighbors.

the changes in the overlay, are urgently required. Furthermore, network conditions have to be monitored to maximize the utilization of available resources, while simultaneously providing an acceptable performance.

The fact that peer resources are heterogeneous poses another challenge. In order to minimize end-to-end delay, the most appropriate peers have to be chosen as sender peers and intermediate peers. Thereby, underlay network optimization should be taken into consideration and global control overhead should be kept low, to keep the system scalable.

An additional challenge is to select an appropriate coding scheme to overcome the effects of error-prone transmissions. The scheme has to be flexible due to constant churn and resource heterogeneity. For media coding and FEC the encoding and decoding efficiency is also of importance. Thereby, encoding efficiency is significant especially for live streaming, since the stream has to be decoded immediately. As already mentioned above, the overhead for encoding and decoding should be kept low, to achieve low costs and scalability, therefore, efficient video codecs are an important part of the streaming system.

Free-riding should also be prevented; otherwise the system would not be scalable for large user crowds. Moreover, content availability is a challenge for streaming. A streaming system should make sure, that popular as well as unpopular content is available. To achieve this, caching and replication could be an option.

6.2 Combination of Different Mechanisms

In addition to the already mentioned challenges inherent to P2P streaming, new challenges arise when combining different resilience mechanisms. In this regard, one should make sure that there is not too much management overhead. Overhead is incurred through the creation and maintenance of the overlay topology. Maintenance overhead includes, for example, peer selection, content discovery, and scheduling.

While the overhead of a single resilience mechanism might not have a great impact on scalability, the combined amount of overhead from multiple mechanisms could degrade scalability significantly and, therefore, render the streaming system useless.

Another challenge arises when layered coding mechanisms such as SVC are deployed [92]. Since decoding is not possible if the base layer is not received on time, the scheduling mechanism should ensure that the transmission of the base layer is scheduled with higher priority, perhaps even with replication over different streaming paths.

When streaming video data over multi-paths (as done in mesh and multi-tree approaches), the scheduler should be aware of and make decisions regarding the proper amount of redundancy for transmission while using FEC-MDC and FEC. Too much transmitted redundant data leads to wasted resources and maybe even connections. Further evaluation of suitable error correction parameters should be performed depending on the level of perturbations in the network.

If retransmission is deployed, it is crucial that the time required for transmission is as small as possible. Therefore, the topology respectively peer selection mechanisms should be chosen properly. For example, neighbors should be chosen with awareness of the underlay to minimize transmission delays. Moreover, heterogeneous peer resources can be challenging in the context of scheduling and retransmission. An issue might evolve if the path with the smallest delay is the one with the smallest available bandwidth. In this case the applied scheduling mechanism should make sure that the piece, which has to be retransmitted, is transmitted immediately if it is an important piece. In case the piece is not of high importance, the algorithm has to consider whether the piece can be dropped, especially if either the following pieces are more important or if it is likely that the piece would not arrive on time at the receiver anyway.

6.3 Research Gap

Although there exists a plethora of research, combinations of different resilience mechanisms are not tackled thoroughly enough. There exist some systems, in which several approaches are combined such as AnySee [64], which possesses an underlay-aware mesh topology and deploys multi-path streaming, or PeerStreaming [42], which is based on a tree topology and utilizes ARQ and FEC. But more extensive research about different possible combinations together with an extensive simulation study should be performed to enable better comparisons regarding performance and resilience of the different combinations. Moreover, it is also important to identify the weaknesses of specific combinations.

There exist some interesting directions for possible research. For example, one could build a mesh-based system, which utilizes multi-path streaming where additional coding schemes would be used. It is important to compare and simulate different coding schemes in this context to see what effects on resilience can be obtained. Furthermore, it would be interesting to optimize scheduling in the context of mesh-based or hybrid approaches to identify the tradeoffs between having a resilient and efficient streaming system. Compared to sim-

ulations and formal analysis, real deployment scenarios would yield more realistic results, for determining which combinations are most promising, and what combinations suffer from specific drawbacks. This would aid in choosing the most appropriate combinations for specific deployment scenarios, such as live streaming or VoD.

References

1. Zhang, X., Liu, J., Li, B., Yum, T.P.: CoolStreaming/DONet: a Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In: INFOCOM, IEEE (2005)
2. Huang, C., Li, J., Ross, K.W.: Can Internet Video-on-demand be Profitable? In: ACM SIGCOMM, ACM (2007)
3. Liu, Y., Guo, Y., Liang, C.: A Survey on Peer-to-Peer Video Streaming Systems. Peer-to-Peer Networking and Applications **1**(1) (2008) 18–28
4. Jurca, D., Chakareski, J., Wagner, J.P., Frossard, P.: Enabling Adaptive Video Streaming in P2P Systems. IEEE Communications Magazine **45**(6) (2007) 108–114
5. Pussep, K., Oechsner, S., Abboud, O., Kantor, M., Stiller, B.: Impact of Self-Organization in Peer-to-Peer Overlays on Underlay Utilization. In: International Conference on Internet and Web Applications and Services (ICIW). (2009)
6. Abboud, O., Pussep, K., Kovacevic, A., Steinmetz, R.: Quality adaptive peer-to-peer streaming using scalable video coding. Wired-Wireless Multimedia Networks and Services Management (2009)
7. Pussep, K., Abboud, O., Gerlach, F., Steinmetz, R., Strufe, T.: Adaptive Server Allocation for Peer-assisted VoD. In: International Parallel and Distributed Processing Symposium (IPDPS), IEEE (2010)
8. Laprie, J.C.: From Dependability to Resilience. In: International Conference on Dependable Systems and Networks (DSN), IEEE/IFIP (2008)
9. Brinkmeier, M., Fischer, M., Grau, S., Schaefer, G., Strufe, T.: Methods for Improving Resilience in Communication Networks and P2P Overlays. PIK – Praxis der Informationsverarbeitung und Kommunikation **32** (2009) 64–78
10. Jiang, X., Dong, Y., Xu, D., Bhargava, B.: GnuStream: A P2P Media Streaming System prototype. In: International Conference on Multimedia and Expo (ICME), IEEE (2003)
11. Wu, C., Li, B.: rStream: Resilient and Optimal Peer-to-Peer Streaming with Rateless Codes. IEEE Transactions on Parallel and Distributed Systems **19**(1) (2008) 77–92
12. Cascella, R.: The "Value" of Reputation in Peer-to-Peer Networks. In: Consumer Communications and Networking Conference (CCNC), IEEE (2008)
13. Abboud, O., Kovacevic, A., Graffi, K., Pussep, K., Steinmetz, R.: Underlay Awareness in P2P Systems: Techniques and Challenges. In: International Parallel and Distributed Processing Symposium (IPDPS), IEEE (2009)
14. Liu, J., Rao, S.G., Li, B., Zhang, H.: Opportunities and challenges of Peer-to-Peer Internet Video Broadcast. Proceedings of the IEEE **96**(1) (2008) 11–24
15. Ramamurthy, B., Ghoshal, J., Xu, L.: Variable Neighbor Selection in Live Peer-to-Peer Multimedia Streaming Networks. Technical report, Department of Computer Science and Engineering, University of Nebraska-Lincoln (2007)
16. Ghoshal, J., Xu, L., Ramamurthy, B., Wang, M.: Network Architectures for Live Peer-to-Peer Media Streaming. Technical report, Department of Computer Science and Engineering, University of Nebraska-Lincoln (2007)
17. Gifford, D.K., Johnson, K.L., Kaashoek, M.F., Jr., J.W.O.: Overcast: Reliable Multicasting with an Overlay Network. In: USENIX Symposium on Operating Systems Design and Implementation (OSDI). (2000)
18. Chu, Y.H., Rao, G., Zhang, H.: A Case for End System Multicast. In: SIGMETRICS. (2000)
19. Liao, R., Yu, S., Yu, J.: Synchronization-based Overlay Construction for Resilient P2P Streaming. In: World Congress on Intelligent Control and Automation (WCICA). (2008)
20. Padmanabhan, V., Wang, H., Chou, P.: Supporting Heterogeneity and Congestion Control in Peer-to-Peer Multicast Streaming. In: International Peer to Peer Symposium (IPTPS). (2004)
21. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: SplitStream: High-bandwidth Multicast in Cooperative Environments. In: Symposium on Operating Systems Principles (SOSP), ACM (2003)
22. Padmanabhan, V.N., Wang, H.J., Chou, P.A.: Resilient Peer-to-Peer Streaming. In: IEEE International Conference on Network Protocols (ICNP). (2003)
23. Medard, M., Finn, S.G., Barry, R.A., Gallager, R.G.: Redundant Trees for Preplanned Recovery in Arbitrary Vertex-redundant or Edge-redundant Graphs. IEEE/ACM Transactions on Networking **5**(7) (1999) 641–652
24. Magharei, N., Rejaie, R.: PRIME: Peer-to-Peer receiver-driven mesh-based Streaming. In: IEEE INFOCOM. (2007)
25. Magharei, N., Rejaie, R., Guo, Y.: Mesh or Multiple-tree: A comparative study of Live P2P Streaming approaches. In: IEEE INFOCOM. (2007)
26. Venkataraman, V., Yoshida, K., Francis, P.: Chunkyspread: Heterogeneous unstructured tree-based peer to peer Multicast. In: IEEE International Conference on Network Protocols (ICNP). (2006)
27. Wang, F., Xiongand, Y., Liu, J.: mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast. In: IEEE International Conference on Distributed Computing Systems (ICDCS). (2007)
28. Asaduzzaman, S., Qiao, Y., Bochmann, G.V.: CliqueStream: An Efficient and Fault-resilient Live Streaming Network on a Clustered Peer-to-Peer Overlay. In: IEEE International Conference on Peer-to-Peer Computing (P2P). (2008)
29. Abdouni, B., Cheng, W., Chow, A.L., Golubchik, L., Lee, W.J., Lui, J.C.: Multi-path Streaming: Optimization and evaluation. In: Multimedia Computing and Networking (MMCN). (2005)
30. Golubchik, L., Lui, J.C.S.: Multi-path Streaming: Is it Worth the Trouble? SIGMETRICS Performance Evaluation Review **30**(3) (2002) 12–14
31. Nguyen, T., Zhakor, A.: Distributed Video Streaming over internet. In: Conference on Multimedia Computing and Networking (MMCN). (2002)
32. Karrer, R., Gross, T.: Multipath Streaming in Best-effort Networks. In: IEEE International Conference on Communications (ICC). (2003)
33. Wei, W., Zakhor, A.: Path Selection for Multi-path Streaming in wireless ad-hoc Networks. In: International Conference on Image In Processing (ICIP), IEEE (2006)
34. Apostolopoulos, J., Wong, T., Tan, W., Wee, S.: On multiple description streaming with content delivery networks. In: IEEE INFOCOM. (2002)
35. Golubchik, L., Lui, J.C.S., Tong, T.F., Chow, L.H., W. J. Lee, G.F., Anglano, C.: Multi-path continuous media streaming: What are the benefits? Technical Report CS-TR-2002-01 (2002)
36. Ghareeb, M.: About Multiple Paths Video-Streaming: State of the Art. Technical report, INRIA (2008)

37. Setton, E., Baccichet, P., Girod, B.: Peer-to-Peer Live Multicast: A Video Perspective. Proceedings of the IEEE **96**(1) (2008) 25–38
38. Setton, E., Noh, J., Girod, B.: Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming. In: Workshop on Advances in Peer-to-Peer Multimedia Streaming at ACM Multimedia. (2005)
39. Mavlankar, A., Noh, J., Baccichet, P., Girod, B.: Optimal Server Bandwidth Allocation for Streaming Multiple Streams via P2P Multicast. In: IEEE Workshop on Multimedia Signal Processing (MMSP). (2008)
40. Frossard, P., de Martin, J.C., Civanlar, R.: Media streaming with network diversity. In Proceedings of the IEEE **96**(1) (2008) 39–53
41. Rizzo, L.: Effective Erasure Codes for Reliable Computer Communication Protocols. ACM SIGCOMM Computer Communication Review **27** (1997) 24–36
42. Li, J., Cui, Y., Chang, B.: Peerstreaming: design and implementation of an on-demand distributed streaming system with digital rights management capabilities. Multimedia Systems **13** (2007) 173–190
43. Guo, Q., Zhang, Q., Zhu, W., , Zhang, Y.Q.: Sender-adaptive and Receiver-driven Video Multicasting. In: International Symposium on Circuits and Systems (ISCAS), IEEE (2001)
44. Rhee, I.: Error Control Techniques for Interactive Low-bitrate Video Transmission over the Internet. In: ACM SIGCOMM. (1998)
45. Dan, G., Chatzidrossos, I., Fodor, V., Karlsson, G.: On the Performance of Error-Resilient End-Point-Based Multicast Streaming. In: IEEE International Workshop on Quality of Service (IWQoS). (2006)
46. Kaune, S., Pussep, K., Tyson, G., Mauthe, A., Steinmetz, R.: Cooperation in P2P Systems through Sociological Incentive Patterns. In: 3rd International Workshop on Self-Organizing Systems (IWSOS), Springer (2008) 10–22
47. Habib, A., Chuang, J.: Incentive Mechanism for Peer-to-Peer Media Streaming. In: IEEE International Workshop on Quality of Service (IWQoS). (2004)
48. Piatek, M., Krishnamurthy, A., Venkataraman, A., Yang, R., Zhang, D., Jaffe, A.: Contracts: Practical contribution incentives for p2p live streaming. In: In USENIX NSDI. (2010)
49. Abboud, O., Zinner, T., Lidanski, E., Pussep, K., Steinmetz, R.: StreamSocial: A P2P Streaming System with Social Incentives. In: IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM). (2010)
50. Cohen, B.: Incentives Build Robustness in BitTorrent. In: 1st Workshop on Economics of Peer-to-Peer Systems. (2003)
51. Liu, Z., Shen, Y., Panwar, S.S., Ross, K.W., Wang, Y.: Using Layered Video to Provide Incentives in P2P Live Streaming. In: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV. P2P-TV '07, New York, NY, USA, ACM (2007) 311–316
52. Li, H., Clement, A., Marchetti, M., Kapritsos, M., Robinson, L., Alvisi, L., Dahlin, M.: FlightPath: Obedience vs Choice in Cooperative Services. In: OSDI 2008. (2008)
53. Ruohomaa, S., Kutvonen, L., Koutrouli, E.: Reputation Management Survey. In: International Conference on Availability, Reliability and Security (ARES). (2007)
54. S. Song, K. Hwang, R.Z., Kwok, Y.K.: Trusted P2P Transactions with Fuzzy Reputation Aggregation. IEEE Internet Computing **6**(9) (2005) 24–34
55. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. IEEE Transactions on Knowledge and Data Engineering **7**(16) (2004) 843–857
56. S. Lee, R.S., Bhattacharjee, B.: Cooperative Peer Groups in NICE. In: IEEE INFOCOM. (2003)
57. Yu, B., Singh, M.P.: Distributed Reputation Management for Electronic Commerce. Computational Intelligence **18**(4) (2002) 535–549
58. Kaune, S., Tyson, G., Pussep, K., Mauthe, A., Steinmetz, R.: The Seeder Promotion Problem: Measurements, Analysis and Solution Space. In: IEEE International Conference on Computer Communications and Networks (ICCCN). (2010)
59. Zhang, X.Y., Zhang, Q., Zhang, Z., Song, G., Zhu, W.: A Construction of Locality-aware Overlay Network: mOverlay and its Performance. IEEE Journal on Selected Areas in Communications **1**(22) (2004) 18–28
60. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks. In Schiper, A., Shvartsman, A.A., Weatherspoon, H., Zhao, B.Y., eds.: Future Directions in Distributed Computing. Springer (2003) 103–107
61. Yu, Y., Lee, S., Zhang, Z.L.: Leopard: A Locality Aware Peer-to-Peer System with no Hot Spot. In: International Conferences on Networking. (2005)
62. Liang, J., Nahrstedt, K.: DagStream: Locality Aware and Failure Resilient Peer-to-Peer Streaming. In: SPIE/ACM Multimedia Computing and Networking (MMCN). (2006)
63. Tran, D.A., Hua, K.A., Do, T.: ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In: IEEE INFOCOM. (2003)
64. Liao, X., Jin, H., Liu, Y., Ni, L.M., Deng, D.: AnySee: Peer-to-Peer Live Streaming. In: IEEE INFOCOM. (2006)
65. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. IEEE Transactions on Circuits and Systems for Video Technology **17**(9) (2007) 1103–1120
66. Baccichet, P., Schierl, T., Wiegand, T., Girod, B.: Low-delay Peer-to-Peer Streaming Using Scalable Video Coding. In: Packet Video Workshop. (2007)
67. Abboud, O., Zinner, T., Pussep, K., Al-Sabea, S., Steinmetz, R.: On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems. In: ACM Multimedia Systems 2011, ACM (2011)
68. Cui, Y., Nahrstedt, K.: Layered Peer-to-Peer Streaming. In: International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV). (2003)
69. Shen, Y., Liu, Z., Panwar, S.S., Ross, K.W., Wang, Y.: Streaming Layered Encoded Video Using Peers. In: IEEE International Conference on Multimedia and Expo. (2005)
70. Chakareski, J., Han, S., Girod, B.: Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths. Multimedia Systems **10** (2005) 275–285
71. Nguyen, A., Li, B., Elisassen, F.: Chameleon: Adaptive Peer-to-Peer Streaming with Network Coding. In: IEEE INFOCOM. (2010)
72. Wang, M., Li, B.: R2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming. IEEE Journal on Selected Areas in Communications **25**(9) (2007) 1655–1666
73. Liu, Z., Wu, C., Li, B., Zhao, S.: UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding. In: IEEE INFOCOM. (2010)
74. Shojania, H., Li, B.: Tenor: Making Coding Practical from Servers to Smartphones. In: ACM International Conference on Multimedia. (2010)
75. Nafaa, A., Taleb, T., Murphy, L.: Forward Error Correction Strategies for Media Streaming over Wireless Networks. Communications Magazine, IEEE **46**(1) (2008) 72–79
76. Schuman, C.D., Plank, J.S.: A Performance Comparison of Open-Source Erasure Coding Libraries for Storage Applications. Technical Report UT-CS-08-625, Department of Elec-

trical Engineering and Computer Science, University of Tennessee (2008)
77. Reed, I.S., Solomon, G.: Polynomial Codes Over Certain Finite Fields. Journal of the Society for Industrial and Applied Mathematics (8) (1960) 300–304
78. Johannesson, R., Zigangirov, K.S.: Fundamentals of Convolutional Coding. IEEE Press, Series on digital and mobile communication (1999)
79. Gallager, R.G.: Low-Density Parity-Check Codes. IRE Transactions on Information Theory **8**(1) (1962) 21–28
80. Schmidt, G., Sidorenko, V.R., Bossert, M.: Collaborative Decoding of Interleaved Reed-Solomon Codes and Concatenated Code Designs. CoRR (2006)
81. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: International Workshop on Design Issues in Anonymity and Unobservability. (2000)
82. Druschel, P., Rowstron, A.: Past: a large-scale, persistent peer-to-peer storage utility. In: Workshop on Hot Topics in Operating Systems (HotOS). (2001)
83. Strufe, T.: A Peer-to-Peer-based Approach for the Transmission of Live Multimedia Streams. PhD thesis, TU Ilmenau (2007)
84. Gopalakrishnan, V., Silaghi, B., Bhattacharjee, B., Keleher, P.: Adaptive Replication in Peer-to-Peer Systems. In: IEEE International Conference on Distributed Computing Systems (ICDCS). (2004)
85. Small, T., Liang, B., , Li, B.: Scaling Laws and Tradeoffs in Peer-to-Peer Live Multimedia Streaming. In: ACM International Conference on Multimedia. (2006)
86. Wu, D., Hou, Y.T., Zhu, W., Zhang, Y.Q., Peha, J.M.: Streaming Video over the internet: Approaches and directions. IEEE Transactions on Circuits and Systems for Video Technology **11**(3) (2001) 282–300
87. Miao, Z., Ortega, A.: Proxy Caching for Efficient Video Services over the Internet. In: International Packet Video Workshop (PVW). (1999)
88. Kermode, R.G.: Smart Network Caches: Localized Content and Application Negotiated Recovery Mechanisms for Multicast Media Distribution. PhD thesis, MIT, Cambridge (1998)
89. Lin, W.K., Chiu, D.M., Lee, Y.B.: Erasure Code Replication Revisited. In: IEEE International Conference on Peer-to-Peer Computing (P2P). (2004)
90. Liu, T., Nelakuditi, S.: Disruption-tolerant Content-aware Video Streaming. In: ACM International Conference on Multimedia. (2004)
91. Jiang, H., Jin, S.: NSYNC: Network Synchronization for Peer-to-Peer Streaming Overlay Construction. In: International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), ACM (2006)
92. Abboud, O., Zinner, T., Pussep, K., Oechsner, S., Steinmetz, R., Tran-Gia, P.: A QoE-Aware P2P Streaming System Using Scalable Video Coding. In: IEEE International Conference on Peer-to-Peer Computing (P2P). (2010)