Signal Processing: *Image Communication* 27 (2012) 457–469

Contents lists available at SciVerse ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Peer-to-peer streaming in heterogeneous environments

Remo Meier*, Roger Wattenhofer

Swiss Federal Institute of Technology, Distributed Computing Group, 8092 Zurich, Switzerland

ARTICLE INFO

Available online 21 February 2012

Keywords:
Peer-to-peer
Streaming
Scalable coding
Erasure coding

ABSTRACT

Peer-to-peer overlay networks are comprised of different kinds of devices, from mobile phones to high-definition televisions. They differ in size, computational power, and Internet access. The design of any peer-to-peer system has to account for such heterogeneous environments. For example, in the context of content delivery systems, the content must be delivered reliably, on time, and in a format suitable for each peer.

This work addresses the heterogeneity and reliability of peers in peer-to-peer streaming applications. It applies lessons learned from distributed hash tables (DHTs) by adopting a prefix-based overlay structure. The flexibility of its neighbor selection policy is exploited to make use of scalable coding and erasure coding schemes, bringing different kinds of peers together in a single overlay network. Thereby, each peer can select the appropriate number of scalable coding layers to obtain content in a suitable format. The prefix-based nature further allows efficient content distribution with low-delay, simple maintenance, strong connectivity, and quick adaption to changing conditions; making the proposed algorithms desirable for real-world use, for both peer-to-peer live and on-demand streaming.

1. Introduction

Peer-to-peer technology is an appealing paradigm for the distribution of audio and video content. It allows utilizing the resources of participating peers to overcome the shortcomings of more centralized, server-based approaches. More recently, there is a trend toward applications that deliver content in real-time, such as peer-to-peer streaming applications. At the same time, we witness a variety of devices from mobile phones and tablet computers up to high-definition three-dimensional televisions gaining access to the Internet. The design of any peer-to-peer system has to consider this heterogeneous nature of peers.

For file-sharing applications, the capabilities of peers only influence the average download time. In contrast, streaming applications face the challenge of delivering content with strict a playback deadline; content not delivered on time is of no use to a peer, and is discarded. This makes the design of any peer-to-peer streaming system more intricate. It is practically impossible to distribute content encoded in a single format, i.e., with a given resolution and bitrate. Peers may lack the bandwidth to sustain a stream. Weaker peers may be unable to process the incoming content. Action sequences in a video stream can lead to bursts of packets. Mobile devices lack the display to output high definition content. And changing network conditions may abruptly alter the situation of individual peers.

In this work we present novel techniques to cope with heterogeneous peers. A structured yet flexible overlay network is able to accommodate arbitrary heterogeneous sets of peers, and allows the delivery of both live and on-demand streams. The overlay employs a prefixed-based routing policy, similar to distributed hash tables, to gain desirable properties, such as an efficient distribution with low delay, robustness to churn, and guaranteed connectivity among all

* Corresponding author.
E-mail addresses: remmeier@tik.ee.ethz.ch (R. Meier), wattenhofer@tik.ee.ethz.ch (R. Wattenhofer).

0923-5965/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2012.02.008

peers with a logarithmic overlay diameter. Scalable audio and video coding schemes (whereas content is partitioned into multiple coding layers) and erasure coding schemes complement the overlay structure to address the varying capabilities and reliability of peers. In contrast to other work, a *single* connected prefix-based overlay network is sufficient to distribute all coding layers. This eases the implementation, provides better robustness, and allows peers to more quickly adapt to changing network conditions.

The subsequent Section 2 reviews techniques to approach the heterogeneous nature of peers and studies related literature. Scalable coding schemes are presented in Section 3. Section 4 presents our overlay structure and content distribution mechanism. While the focus is on live streaming protocols, the used techniques are also applicable to on-demand and live/on-demand hybrid streaming protocols. Finally, Section 5 evaluates the proposed mechanisms and Section 6 concludes this paper.

2. Related work

Peer-to-peer live streaming protocols are mainly categorized according to the topology maintained among the peers or, equivalently, the neighbor selection algorithms the peers employ. Simple multicast systems are based on overlay *trees* [3,7,29]. While tree topologies are conceptually simple, there are rather serious drawbacks which render such systems inefficient. For example, resources are wasted as the leaves of such a tree do not contribute anything to the system, while inner nodes having two children need to upload at twice the bitrate of the stream. The fragile tree structure is not resistant to peer failures or churn, and a peer is limited by its weakest predecessor in the tree. While maintaining several trees [2] improves the robustness of a system, each tree can break individually, and the overhead potentially increases as more trees have to be repaired continuously (and concurrently).

Since a rigidly structured overlay requires permanent maintenance, care has to be taken not to burden the individual peers. Therefore, unstructured overlays have been favored over structured overlays, and various protocols based on unstructured overlays have been proposed, e.g., *CoolStreaming*|*DONet* [31], *Chainsaw* [18] and *GridMedia* [14]. Typically in unstructured overlays, peers have to *notify* neighboring peers about available blocks of data, and peers that are interested in obtaining these blocks must explicitly *request* them before any data is exchanged, because there is no structure in the network that can be used to disseminate data. Unstructured overlays are considered more robust, whereas they have the disadvantage that notifying peers and subsequently requesting data blocks potentially results in long delays before any data is exchanged. If further optimizations are applied (such as favoring connections among near-by peers) there is the additional concern that unstructured overlays fall apart due to the lack of structure and the formation of clusters.

In the literature, there are four main techniques to fix the lack of guarantees and the heterogeneous nature of peers: stream switching, source coding, multiple description coding (MDC), and scalable coding. Each coding technique has its own merits.

Stream switching encodes the content at different bitrates. A peer chooses a bitrate for download based on its capabilities and available bandwidth. The peer may later switch between bitrates to adapt to changing network conditions. For video streams, switching takes place at regular key frames or dedicated switching frames [9]. There are a variety of different implementations. More recently, HTTP Live Streaming[1] (also referred to as HLS) has been submitted to the IETF for standardization and is gaining in popularity due to its simplistic HTTP-based design. Stream switching works well in server-based environments, but it faces challenges when applied in peer-to-peer systems. Streams with different bitrates are independent of one another. Accordingly, peer-to-peer systems have to partition the peers into groups based on the chosen bitrate and thereby may separate near-by peers. Switching between streams is prohibitively expensive; peers have to replace connections to neighboring peers and the contents of their buffers. Furthermore, stream switching is not resilient to missing data blocks. Peers would have to choose a sufficiently low quality to ensure a timely delivery of all blocks at all times.

Erasure codes, such as Reed–Solomon codes [21], LT codes [13], and Raptor codes [25], allow the generation of n coded blocks from k original data blocks with $n > k$, whereas any $k+\epsilon$ coded blocks with $\epsilon \geq 0$ allow the reconstruction of the original data blocks. For a perfect erasure coding scheme, it holds that $\epsilon = 0$. However, most schemes trade the optimal message complexity for a lower computational complexity with $\epsilon > 0$. In the context of peer-to-peer streaming and file sharing systems, erasure codes are used to implement *source coding*. The source performs an erasure coding of the content before distributing it in its swarm. Erasure coding schemes allow users to cope with lost data blocks caused by unreliable peers and network links. However, source coding cannot cope with heterogeneous peers. Any $k+\epsilon-1$ coded blocks are statistically independent of the original blocks. Accordingly, the download of an insufficient number of blocks leads to no useful information at all.

Finally, the (related) coding techniques *multiple description coding* (*MDC*) [5] and scalable coding [23]: Both coding schemes produce pictures and audio samples from subsets of data blocks. The size of the subset determines the quality. While MDC decodes any subset of data blocks, scalable coding enforces a pyramid-like structure: data is partitioned into a base layer and multiple enhancement layers. The base layer is mandatory to obtain an initial low quality result. The download of additional enhancement layers gradually improves the quality, but only if the layers beneath are available as well. While MDC is more flexible, scalable coding is more efficient. With a pyramid-like structure, scalable coding schemes more closely resemble current state-of-the-art coding schemes such as H.264 [30]. Simply speaking, H.264 employs two mechanisms. First, the current picture

[1] http://tools.ietf.org/html/draft-pantos-http-live-streaming-06

is estimated from previous (reference) pictures by encoding a set of motion vectors into the bitstream. Second, it transforms the estimation errors into a frequency domain, quantizes the frequency coefficients, and encodes the quantized frequency coefficients into the bitstream. A pyramid-like structure allows efficient implementation of scalable coding schemes by gradually improving, for example, the quantized frequency coefficients. In contrast, MDC has to use more elaborate techniques to gain its properties, rendering an implementation expensive in terms of computational complexity and bitrate. The close relation of scalable and regular coding schemes further allows the reuse of many existing software and hardware-based optimizations.

The properties of MDC and scalable coding make these coding schemes desirable for use in heterogeneous peer-to-peer systems. Multi tree-based systems like SplitStream [2] frequently refer to MDC to mask problems in the employed distribution trees inflicted by selfish behavior, churn, and congestion. A more detailed study of MDC in the context of tree-based systems is given in [20]. Reasonable playback quality is achieved whenever a sufficient number of trees provide data blocks. Scalable coding schemes garner less attention because they address only the heterogeneous nature of peers, not their reliability. A missing key frame in the base layer prevents the use of any downloaded data from any layer. Nevertheless, considering their close relation to existing schemes and their superior efficiency, and taking into account that bandwidth is the most precious resource in content delivery systems, scalable coding is the more suitable approach.

Source coding, applied to each scalable coding layer, may complement scalable coding to gain the needed resilience against unreliable peers. In [15,16], the authors propose a similar approach by employing network coding in addition to scalable coding. While network coding is more flexible than source coding, it entails significant computational overhead for both the reconstruction of the original data blocks and the necessary techniques to authenticate network coded blocks.

Baccichet et al. [1] augment the *Stanford Peer-to-Peer Multicast* (SPPM) protocol [17] with scalable video coding support. Peers are organized into multicast trees. Unlike SplitStream, blocks of different coding layers are distributed within the same trees. To account for the pyramid-like structure of scalable content, more powerful peers are positioned closer to the source and peers drop higher (optional) coding layers if they are not able to forward all incoming data to their children. Unfortunately, the proposed protocol assumes a static setting wherein peers have a known upload capacity. Changing the download rate of a peer requires restructuring the distribution trees. The received download rate of a peer is further limited by the *weakest* predecessor. And peers have an incentive to move close to the source to obtain the best possible quality, which can promote selfish behavior.

This work advocates the use of structured overlays based on a *prefix-routing* neighbor selection policy [19,22]. The prefix-based nature allows an efficient content distribution with low delay, similar to tree-based systems, while maintaining flexible neighbor selection and content exchange policies, similar to unstructured overlays. The flexibility further lets peers make use of scalable and erasure coding schemes. In doing so, arbitrary heterogeneous sets of peers can join the same overlay network, wherein peers can choose the number of consumed coding layers. The prefix-based structure ensures that the overlay remains connected regardless of the interest of peers and the applied optimizations.

3. Overview of scalable coding

There are scalable coding schemes for both video and audio content. Audio content usually has a bitrate from 32 Kbps to 2 Mbps. In contrast, the bitrate of video content ranges from 192 Kbps to 40 Mbps. Clearly, video content dominates in terms of bit complexity. Accordingly, the main focus of this work is on scalable video coding schemes. Scalable audio coding becomes necessary to support low-end devices at low bitrates.

The subsequent subsections discuss scalable video in the context of H.264, which is, at the time of this writing, considered the state-of-the-art video coding scheme. SVC is the scalable video extension of H.264. The reader is referred to [23,30] for a more detailed description. Scalable video coding allows scaling video content in three dimensions: spatial, quality, and temporal scalability. Accordingly, each data block carries a spatial, a quality, and a temporal index. Three-dimensional video, employing separate pictures for different viewing angles, can be considered being a fourth dimension of scalability.

3.1. Spatial scalability

Spatial enhancement layers increase the resolution of a video. Typically, every enhancement layer doubles both the width and height. Three or four spatial layers cover a variety of devices from mobile phones to high-definition televisions. H.264 uses macroblocks 16×16 pixels in size that may further be subdivided into 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 blocks. For this reason, spatial scalability with a factor two fits in naturally when scaling macroblocks from lower layers. A sample screenshot with three spatial layers is depicted in Fig. 1.

Fig. 1. Spatial scalability with three spatial layers. The resolution doubles with each layer.

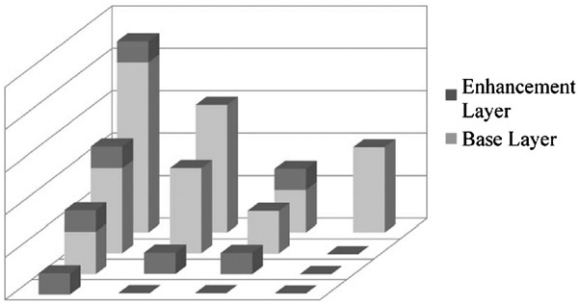

Fig. 2. Refinement of the 16 DCT frequency coefficients of a 4×4 pixel block by a quality enhancement layer. In this example, the base layer quantizes the DCT coefficients to five different values. The enhancement layer doubles the quality of the quantization by allowing 10 different values.

3.2. Quality scalability

Quality enhancement layers improve upon the quality by refining the quantized frequency coefficients (see Fig. 2). The resulting pictures are sharper with fewer artifacts, while the resolution stays the same. Typically two, but not more than three, quality layers should be used. Otherwise, the quantization becomes too coarse to give an acceptable quality to lower layers. In such a case, it is more reasonable to change the resolution by introducing additional spatial layers. The bitrates of quality layers are usually chosen to match the bitrate of all the layers underneath, i.e., doubling the total bitrate. A sample screenshot with four quality layers is depicted in Fig. 3.

3.3. Temporal scalability

Temporal scalability adjusts the frame rate of a video. The motion mechanism of H.264 estimates the current frame from one or more reference frames. By carefully selecting reference frames, it is possible to skip some (non-referenced) frames altogether. Fig. 4 depicts an example with four temporal layers. The decoder may skip up to seven frames by only downloading the temporal base layer. However, the exponential reference structure forces the encoder to encode changes, like the change from one scene to another, multiple times for each temporal layer. A more efficient approach employs bi-predictive frames (B-Frames) whereby the encoder reorders frames before the encoding process; a frame may reference frames from both the past and the future, avoiding the aforementioned issue of redundant coding. While the number of frames grows exponentially with every temporal enhancement layer, temporal layers are of about equal size because the distance to reference frames becomes exponentially smaller.

Temporal scalability serves two purposes. Besides allowing the skipping of frames without breaking the decoding loop, it may complement quality scalability, known as *medium grained* quality scalability. The download of all enhancement frames is not mandatory. The download of enhancement frames with a low temporal index gives an initial slight increase in quality. Clients are not constrained by fixed coding layers and can more efficiently adapt to the available bitrate. The exponential reference structure of temporal scalability ensures that small coding errors inflicted by missing blocks do not propagate and accumulate over a large number of frames.

Fig. 3. Quality scalability with four quality layers. The picture quality increases with each layer; the resolution remains the same. The H.264 quantization parameter QP is decreased by 6 with each layer.

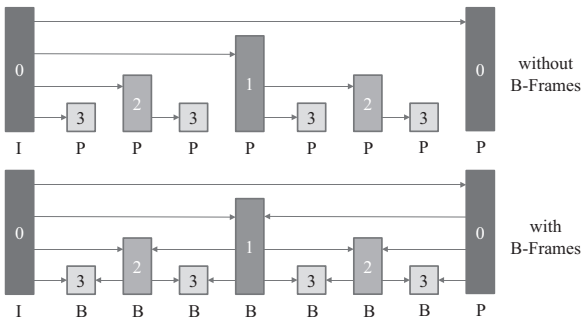

Fig. 4. Reference structure of video frames when temporal scalability is used. The numbers denote the temporal index of a frame. One may or may not place I-frames in a dedicated temporal layer.

3.4. Example

Various configurations combining the three dimensions of scalability are feasible. A configuration may use three spatial layers, i.e., three different resolutions. If the base layer has a resolution of 480×270 pixels then the first enhancement layer has a resolution of 960×540 pixels and the second enhancement layer has a resolution of 1920×1080 pixels. Each resolution may be further partitioned into two quality layers, creating six layers in total. Both quality layers are of about equal size. Seven temporal layers allow skipping up to 63 frames and smoothly adjusting the quality scalability.

3.5. Audio scalability

Scalable audio coding works in a similar fashion by adapting the number of channels, the sample rate, and the quality. Scalable audio coding has, however, no equivalent to the temporal scalability of scalable video coding. Scalable audio coding schemes are implemented, for example, based on AAC, a state-of-the-art audio coding scheme.[2]

4. Protocol

In the following, an overview of our techniques to design a peer-to-peer streaming protocol suitable for heterogeneous environments is given. The focus is on live streaming systems, but the necessary changes for on-demand and hybrid systems are outlined as well. The proposed techniques are independent from the underlying transport mechanism and are applicable to both UDP and TCP-based systems. Our implementation makes use of both UDP and TCP to maximize connectivity among peers in the presence of NAT devices, firewalls, and proxy servers.

First, the protocol defines a data structure to hold scalable content. Second, it specifies the interconnections among peers, i.e., an overlay structure. The overlay is inspired by the structured topologies of *distributed hash tables* (DHTs) which guarantee connectivity and a logarithmic diameter. The flexibility of the neighbor selection policy is exploited to account for the different kinds of peers and additional factors which influence performance, for instance, bandwidth requirements and latency constraints. The topology aims at being resilient during churn and massive correlated failures. Third, the protocol specifies how data is distributed in the overlay network. It introduces a combination of fast push operations and robust pull operations. Fresh data blocks are pushed directly to a fraction of the peers without preceding requests, in order to fuel the subsequent pull-based exchanges and to achieve low latencies in the swarm. The content distribution honors the pyramid-like structure of scalable content; delivering the appropriate number of coding layers to the individual peers.

[2] See http://www.iis.fraunhofer.de/

4.1. Data structure

Our protocol makes use of both source coding and scalable coding as argued in the related work section. Scalable coding partitions the audio and video content into multiple coding layers to serve different peers at different bitrates. Accordingly, it is used to address the heterogeneity of peers, but not their reliability. A missing key frame in the base layer makes the use of any downloaded content impossible. Conversely, erasure coding allows the reconstruction of the original k blocks from $k+\epsilon$ out of n coded blocks, but any $k+\epsilon-1$ coded blocks usually give no information about the original blocks.

Scalable audio and video frames have a variable size from a few hundred bytes to several hundred kilobytes and carry a spatial, quality, and temporal index to denote their position within the coding layers. The presented protocol aims to work with small blocks fitting into the maximum transfer unit (MTU) of typically 1500 bytes. In doing so, peers can immediately forward incoming blocks without having to wait for further fragments. Accordingly, a data block may contain any number of (small) complete media frames and may start, continue, or finish with a fragment of a larger media frame.

Peer maintain multiple buffers for the most recent blocks. Blocks are discarded if they reach some predefined age. There is a dedicated buffer for the base layer and each quality or spatial enhancement layer. Separate buffers ease the exchange of blocks from subsets of layers. Each buffer uses (independent) sequence numbers to address blocks. Accordingly, a spatial index, a quality index, and a sequence number uniquely address a data block from any layer at any given time. The temporal index of a data block is given by the lowest temporal index of a media frame contained within the data block. Data blocks from different temporal layers share a buffer to limit the number of buffers to maintain. In doing so, only six instead of 42 buffers have to be maintained to accommodate three spatial, two quality, and seven temporal layers.

Each data block further carries references to other data blocks. The reference structure is used for the (ordered) delivery of media frames and to prioritize the download of lower layer blocks. Recall that enhancement layer blocks are of no use without the associated base layer blocks. There are two types of references: layer references and temporal references. A layer reference points toward the corresponding data block in the next higher layer with its sequence number. A single bit marks whether the reference follows a spatial or a quality enhancement layer. A temporal reference points to the closest *prior* data block from the *next lower* temporal layer within the same buffer. In case of a data block from the temporal base, i.e., a key frame, the reference points to the prior data block from the temporal base. Only the base layer carries temporal references. Temporal indices of enhancement blocks are estimated by following layer references from lower layers with known temporal indices. If a data block holds several media frames, it may also hold several references. An example is depicted in Fig. 5.

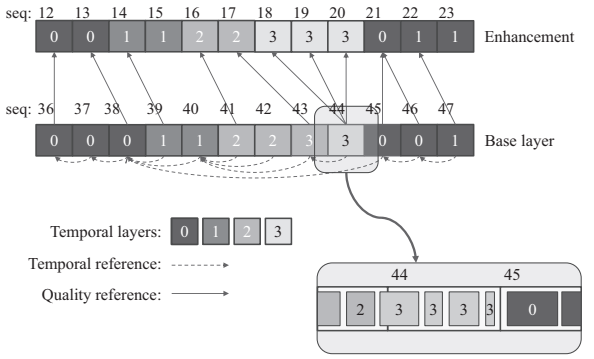

Fig. 5. Example with two buffers holding two quality layers and four temporal layers. Each rectangle represents a data block. Each data block carries a sequence number and some possibly fragmented media frames. For example, the block with sequence number 44 carries four media frames with a temporal index of 3 each (compare with Fig. 4). Part of the first frame is already contained in the previous block. The number and color of a data block denotes its temporal index. The temporal index corresponds to the lowest temporal index of a media frame contained within the data block. References among data blocks are depicted as arrows. For example, the block with sequence number 44 has a temporal reference to the block with the sequence number 43 and multiple quality references to the blocks with the sequence numbers 18, 19, and 20.

4.2. Neighbor selection policy

To maintain desirable overlay properties such as guaranteed small diameter, locality awareness, and robustness to churn, our neighbor selection policy [10] applies techniques used in the context of DHTs, for instance in *Pastry* [22]. This DHT-like overlay is merely used to maintain an efficient streaming topology—no data is stored in the DHT. Each peer carries a d-bit peer identifier and maintains connections to neighboring peers based on shared prefixes of their respective identifiers. These identifiers are later used for prefix-routing, as links to neighbors are stored for different shared prefix lengths. Let β denote the number of bits that can be fixed at a peer to route any message to an arbitrary destination. For $i=\{0,\beta,2\beta,3\beta,\ldots\}$, a peer chooses, if possible, $2^\beta-1$ neighbors whose identifiers are equal in the i most significant bits and differ in the subsequent β bits by one of $2^\beta-1$ possibilities. For example, in the case of $\beta=1$, a peer with identifier 01100 chooses neighboring peers for the identifier prefixes 1, 00, 010, 0111, and 01101 (if such peers exist). If peer identifiers are chosen uniformly at random or in a balanced manner, the length of the longest shared prefix is bounded by $O(\log n)$ in an overlay containing n peers; thus, only $O(\log n(2^\beta-1)/\beta)$ connections need to be maintained. Moreover, every peer reaches every other peer in $O(\log n/\beta)$ hops by repetitively selecting the next hop to fix β more bits toward the destination peer identifier yielding a logarithmic overlay diameter.

Similar to DHTs based on prefix-routing, the proposed solution has the advantage that there is a large choice of neighbors for short prefixes, which means that an optimizing secondary criterion can be used to pick neighbors. With n peers, there are roughly $n/2^{\pi+\beta}$ possibilities to select a neighboring peer for a shared prefix length π. For example, for a shared prefix length $\pi=0$, a peer can choose among half of all the peers to find a suitable neighbor. The possibility of a secondary criterion is used to address the diverse interest of peers in different coding layers. Different kinds of peers may join the same overlay. It is unnecessary to maintain multiple overlays for different coding layers or to position more powerful peers closer to the source – the common approaches to implement scalable coding support – since the prefix-based nature of the proposed overlay already provides the desired flexibility.

To augment the overlay structure with scalable coding support, not only data blocks but also peers carry a spatial, quality, and temporal index, denoted by $i_s(v)$, $i_q(v)$, $i_t(v)$ for peer v. Each peer selects a spatial, quality, and temporal index to reflect the number of layers it is able to download, process, and forward. Thereby, the spatial index $i_s(v)$ has the largest priority. The quality index $i_q(v)$ has medium priority, and the temporal index $i_t(v)$ has the lowest priority.[3] A peer may limit its indices if its display does not output high resolution videos or the computational power is insufficient to decode all layers. Otherwise, a peer's indices reflect its upload capacity, i.e., the number of recently forwarded data blocks, estimated by observing acknowledgements, and packet loss, round trip times. The upload capacity is mapped to the three indices by computing average bitrates of downloaded layers and approximating the bitrates of missing enhancement layers. The proportions between spatial, quality, and temporal layers are fairly constant and allow the estimation of the bitrate of any layer from any other (downloaded) layer. For example, decreasing the quantization parameter QP of H.264 by 6 in a quality enhancement layer roughly doubles the total bitrate, resulting in a quality base layer and a quality enhancement layer of about equal size.

Peers choose neighboring peers not solely based on shared prefixes, but also based on the advertised spatial, quality, and temporal indices as secondary criterion. For each of the $2^\beta-1$ neighbors of a shared prefix, a peer seeks to select a neighbor with similar advertised indices:

1. Preferably, the routing table selects a peer with the same spatial index, quality index, and temporal index.
2. If there is no such peer, the peer with the next higher set of indices is selected.
3. If no such peer is available either, the peer with the highest (lower) set of indices is selected.

For example, the subsequent scenario considers an overlay with $\beta=1$ and two spatial, two quality, and two temporal layers. Let peer v again have identifier 01100. Peer v might be a mobile device with a small screen. For this reason, it limits the download to the spatial base layer, i.e., $i_s(v)=0$. However, the peer has sufficient bandwidth available and downloads all quality and temporal layers, i.e., $(i_s(v),i_q(v),i_t(v))=(0,1,1)$. Peer v again

[3] Depending on the application, the temporal priority may be treated separately from the spatial and the quality priorities.

connects to neighboring peers for the identifier prefixes 1, 00, 010, 0111, and 01101 (if such peers exist). For each prefix, it favors the connections to a peer with the same set of indices. Otherwise, it prefers a peer w with $(i_s(w),i_q(w),i_t(w)) = (1, 0, 0)$. If such a peer does not exist either, it favors (1, 0, 1), (1, 1, 0), (1, 1, 1), (0, 1, 0), (0, 0, 1), or (0, 0, 0) (in this order). Since about half of all the peers carry the prefix 1, a desired peer is likely to be found. For longer prefixes, peer v may have to select a peer with a different set of indices.

The induced overlay structure primarily connects peers with comparable computational power and upload capacity and are able to exchange the desired coding layers. And yet, by obeying the rules of the original DHT-like prefix-based policy, the overlay structure still guarantees connectivity with a logarithmic diameter by maintaining few links to weaker and stronger peers. Regardless of the choice of neighbors, the overlay never falls apart. Peers may continuously refine the selection of neighbors to maintain the best possible set of neighbors. Peers may also choose multiple neighbors for a shared prefix to have a backup should a peer leave or become congested. In most cases, especially for a large number of peers, neighboring peers have equal indices and the remaining flexibility is again used to optimize additional criteria, such as network latency and available bandwidth. For a large number of temporal layers, peers do not have to be too strict when comparing temporal indices. A close temporal index usually suffices if a peer is desirable for other reasons. A peer may also choose to make an exception and ignore the indices altogether for desirable connections, e.g., to close-by peers; such neighboring peers can exchange at the least base layer blocks.

The use of a single overlay among all peers allows peers to smoothly adjust the number of consumed layers by changing the advertised indices. The simplicity of the prefix-based policy and the possibility for backup neighbors ease the implementation and allow neighboring peers to quickly adapt to changing conditions. To bootstrap the formation of the overlay network, peer may make use of centralized tracker to obtain an initial list of peer addresses, similar to protocols like BitTorrent. Later they are free to exchange further peer addresses among each other to maintain the overlay structure in a distributed manner.

4.3. Pushing content

The prime objective of the pushing component is to quickly distribute a data block to a certain number of peers in order to fuel the subsequent pull-based exchanges. As argued before, such a mechanism is needed due to the long delays of purely pull-based approaches; the push phase brings the data block into the vicinity of virtually all peers.

In the following, let, for two peers u and v with identifiers $b_0^u \dots b_{d-1}^u$ and $b_0^v \dots b_{d-1}^v$, where b_i^u and b_i^v denote the ith bit of their respective identifiers, $\ell(u,v) = k$ if $b_j^u = b_j^v$ for all $j \in \{0,\dots,k-1\}$ and $b_k^u \neq b_k^v$. Furthermore, let $\mathcal{N}_v$ be the set of all neighboring peers of v. Let β again denote the number of bits that the prefix routing algorithm fixes at each hop. The source selects 2^β peers from its routing table, if possible, such that the identifiers of any two peers differ in at least one bit of the first β bits. A new block is pushed to these peers along with the information that they must only forward the block to peers with which they share the first β bits of their identifiers. Recursively, upon receiving such a push message with the specified prefix length π that they must not modify, a recipient selects 2^β peers with which it shares the prefix of length π and which differ in at least one bit between the $(\pi+1)$ st and the $(\pi+\beta)$ th bit and so on.

This straightforward approach to pushing on prefix-based overlays has an obvious shortcoming. Assume that $\beta = 2$ and that the source peer has the identifier consisting of only zeros. It will push the block to a peer whose identifier starts with 00, which will in turn forward the block to a peer whose identifier starts with 0000. This peer might then forward the block back to the source again, as the identifier of the source may also start with 000000. Such loops can occur on all paths. A solution to this duplicates problem is to attach a list $\mathcal{L}$ of *critical* predecessors of the induced spanning tree to each pushed block. At each hop, the lists $\mathcal{L}_j$ of critical predecessors are created for all children based on the received list $\mathcal{L}$ and the sender of the block. Any critical predecessor p_i of list $\mathcal{L}$ is added to at most one list $\mathcal{L}_j$ and only if it is still critical for this child v_j, i.e., $\ell(v_j,p_i) \geq \pi+\beta$. Block are subsequently only forwarded to peers if their peer identifiers are not included in their respective lists. The reader is referred to [10] for a more elaborate algorithm that avoids the use of lists.

To honor the diverse interest of peers and the pyramid-like structure of scalable coded content, the push policy follows the first criteria of the neighbor selection policy whereas peers with unsuitable spatial, quality, and temporal indices are to be avoided. While the base layer is exchanged among all peers, higher layers are restricted to peers which have advertised interest in the respective layers. For this purpose, neighbors are selected as children in a push branch based on their advertised indices and the indices of the data block to be forwarded. A child has to satisfy the predicate defined by Algorithm 1. If no child satisfies the predicate, because no further such peers exist that yet have to receive the block in question, the push branch is cut off.[4]

Algorithm 1. push_predicate(v,*data*).

return $i_s(data) < i_s(v)$ **or** $(i_s(data) = i_s(v)$ **and** $i_q(data) < i_q(v))$ **or** $(i_s(data) = i_s(v)$ **and** $i_q(data) = i_q(v)$ **and** $i_t(data) \leq i_t(v))$

Algorithm 2 depicts the final algorithm $\mathcal{ALG}$. It forwards incoming blocks based on the fixed prefix length π and the neighbors' identifiers. Possible children are filtered by the critical predecessor list $\mathcal{L}$ and the advertised spatial, quality, and temporal indices. For the sake of

[4] The overlay network forces peers to have neighbors with undesirable indices to maintain connectivity.

simplicity, the algorithm is depicted with the number of fixed bits set to $\beta = 1$.

Algorithm 2. $\mathcal{ALG}$: push($p,\pi,\mathcal{L},data$) at peer v.

```
1:  S1 := {v' ∈ Nv | ℓ(v',v) = π}
2:  S2 := {v' ∈ Nv | ℓ(v',v) ≥ π+1}
3:  choose v1 ∈ S1 : push_predicate(v1,data)
4:  choose v2 ∈ S2 : push_predicate(v2,data)
5:  for all pi ∈ L do
6:    j := arg max_{j∈{1,2}} ℓ(vj,pi)
7:    if ℓ(vj,pi) ≥ π+β then Lj := Lj ∪ {pi} fi
8:  od
9:  for j = 1,2 do
10:   send push(v,π+β,Lj,data) to vj
11: od
```

The prefix-based, loop-free transmission implies that data is distributed on *induced spanning trees*, which are generally not comparable to structures in which the overlay graph consists of one or more trees which must be used to disseminate data. The overlay structure is still hypercubic, and each packet can induce a different tree on which it is broadcast. At each hop, peers have the flexibility to choose among several neighbors to whom to forward an incoming block. This flexibility allows load-balancing of the upload bandwidth among peers and quick adaptation to changing network conditions such as the joining and leaving of peers and congestion.

The push mechanism does not exhibit a *super-peer*-like structure wherein blocks travel from powerful peers with high indices (super-peers) to weaker peers with lower indices. Peers are considered equal, and the source may send a new block to any peer interested in the block. The same holds for peers forwarding the block. Peers do not gain an advantage for downloading blocks from lower layers by advertising high indices; a useful property when implementing fairness mechanisms. This flexibility of selecting children from any layer may further be used to optimize additional criteria. For example, there may be close-by peers interested in fewer coding layers that are well suited to exchange blocks from the layers in question.

4.4. Pulling content

The push phase is followed by a pull phase whereas peers initiate the transmission of data blocks *explicitly* by collecting notifications from neighboring peers about available data blocks and requesting missing blocks. The push phase is particularly well suited to attain an *initial* distribution of new blocks as it has most flexibility when the fixed prefix lengths are short. To later reach the remaining peers, the selection of children is stricter. Advocating the exchange of blocks in the pull phase by cutting-off branches in the push phase maintains the flexibility among all block exchanges. The push phase may also fail to reach all peers because of malicious behavior, blocks lost on congested links, or inaccurate routing tables, due to the perpetual arrival and departure of peers. Pull operations can be performed efficiently and with small additional delay; the push phase fuels the subsequent pull phase.

Like the push policy, the pull policy has to honor the pyramid-like structure of scalable content; lower layers have to be requested first. The block format of Section 4.1 allocates dedicated buffers for different coding layers, so prioritizing their download is simple. Further prioritizing the download of low temporal layers is more demanding. Recall that temporal layers share a buffer to limit the number of buffers and the associated overhead incurred by periodic notifications. Accordingly, peers first have to learn the temporal indices of blocks within the various buffers to prioritize their download. Peers estimate temporal indices based on already received blocks (from both the push and the pull phase) and the references within the blocks. An estimation mechanism may further account for fragmented frames, estimated block sizes, and block rates.

Prioritizing the download of low temporal layers serves two purposes. First, the *medium grained* scalability of SVC allows the decoding of partially downloaded layers. Second, peers may skip a large number of frames without losing the synchronization in the decoding loop, which otherwise would lead to artifacts until the next key frame is decoded. Such a scenario may occur during a brief burst of lost packets.

The push policy from the previous subsection strictly forbids forwarding blocks to peers with lower than necessary indices to avoid overloading their upload links. In contrast, a peer is free to request blocks from any number of layers regardless of the advertised indices, given that neighbors have sufficient upload capacity. One may apply, for example, a tit-for-tat-based fairness mechanism, such as [4,6,8,11,28,24], to deter peers from consuming resources without contributing. In conjunction with scalable coding, fairness mechanisms allow peers to gradually improve playback quality by contributing more resources themselves. Peers are no longer bound to either download *all* blocks or suffer from unacceptable artifacts due to missing frames that are not obtained on time for playback.

4.5. Source coding

Prefix-based overlays allow peers to maintain backup connections, adapt to congestion, and quickly repair routing tables after other peers join and leave. Nevertheless, there is still a small but non-negligible chance of an error that cuts off a branch in the push phase before the branch reaches a sufficient number of peers. This may also happen because of misbehaving peers. In contrast to scalable coding, source coding allows to address such reliability issues. The download of $k+\epsilon$ out of n source coded blocks is sufficient to reconstruct the original data blocks.

Source coding works well with prefix-based algorithms. Peers have the flexibility to choose among several neighbors to whom to forward an incoming block. This flexibility allows source coded blocks to take different routes to peers and as a consequence at least some blocks likely attain sufficient distribution. Again, it is not necessary to maintain multiple independent overlays, the prefix-based overlay already

provides the needed flexibility. To prevent peers from receiving more than $k+\epsilon$ out of n source coded blocks, individual push branches of source coded blocks are simply cut off earlier than branches of regular data blocks. Peers then obtain missing blocks in the subsequent pull phase. To complement scalable coding and maintain their respective advantages, source coding has to be applied to each scalable coded layer separately.

4.6. On-demand streaming

While the proposed techniques are introduced in the context of peer-to-peer live streaming, they are applicable to on-demand and hybrid streaming systems as well. In the latter case, broadcasted live streams become immediately available as on-demand content. The most notable difference is that peers are no longer interested in the same content at the same time. Some peers may already have obtained the complete file and can act as seeders by caching downloaded content. Other peers have a playback position and are interested in the content at that position. Accordingly, one can introduce the playback position as additional criterion for the neighbor selection policy, similarly to the spatial, quality, and temporal indices. Usually, peers favor neighboring peers based on the incurred network latency to reduce the network load. However, in bandwidth-constrained situations, peers download content from peers with similar playback positions, yielding a linked list-like structure where a single source can deliver a file to an arbitrary number of peers. On-demand streams may not employ the push mechanism (but hybrid systems still can). In doing so, the same system can be used for both live and on-demand streaming with only minor adjustments necessary. Moreover, on-demand streams benefit from the same mechanisms as live streams, such as a single overlay network, simple maintenance, and strong connectivity guarantees.

The proposed architecture further eases the implementation of seeking operations in terms of both locating desirable neighboring peers and quickly obtaining the necessary data blocks to start playback. For example, the DHT-like overlay structure allows the build-up of a distributed index structure to allow the retrieval of peer addresses having already downloaded and cached desired content. The use of scalable and erasure coding shortens the playback delay after seeking. Peers may choose to prioritize the download and playback of the scalable base layer after seeking, saving both download and decoding time. Erasure coding lets peers more aggressively download sought content. Thereby, peers do not have to wait for acknowledgements and retransmissions. Downloading of a sufficient number of arbitrary source coded blocks is sufficient to reconstruct the original data blocks.

Finally, the evaluation section will show that the overlay structure is simple to maintain and repair in the event of joins, leaves, and changes in neighbor preferences (like the playback positions).

5. Evaluation

Emulations are performed in order to study the proposed mechanisms in different environments. The emulator makes use of a real-world implementation executed within a simulated network environment. The emulator runs on a single computer using 64 GB of memory. Since each peer instance merely requires a few kilobytes of memory, we were able to emulate up to 100,000 peers. As in the previous section, the protocol is evaluated with a focus on peer-to-peer live streaming applications.

The evaluation makes use of a scalable video codec that has been implemented as part of the presented system. The codec is based on H.264 and resembles the recently released *SVC* standard [23]. It supports spatial, quality, and temporal scalability to adapt the video resolution, picture quality, and frame rate. Its overhead over a single-layer stream is between 10% and 20% with further optimizations possible.

We start by first evaluating the proposed content distribution mechanism in terms of overlay diameter, latency, and resilience to congestion. Later, we show that peers are able to exploit the temporal structure of encoded pictures to optimize the use of the available bandwidth. With limited download capacity, peers favor the download of blocks from low temporal layers and, in particular, key frames. Subsequently, a tit-for-tat policy is applied to add fairness among a heterogeneous set of peers. While resource-rich peers are able to obtain all blocks on time for playback, weaker peers cannot download significantly more than they contribute. Nevertheless, the weaker peers successfully obtain the lower coding layers for playback. The associated scalable coding overhead is then put in perspective by analyzing gains in topology awareness. Finally, resilience to churn is studied.

5.1. Overlay diameter

The diameter is a key property of any overlay structure. It specifies the maximum number of hops necessary to reach any peer from any other peer. Given the prefix-based nature of the proposed algorithms, the diameter is given by $\log n/\beta$ as it is possible to fix at least β bits with each hop's identifier to get closer to the destination identifier. Most importantly, a bounded diameter ensures that the overlay never falls apart. Regardless of the choice of neighbors for shared prefixes, the overlay remains connected.

The hop count, like the diameter, grows logarithmically with the number of peers. Fig. 6 depicts the average number of hops taken by a data block to reach a peer for 100 up to 100,000 peers. The logarithmic hop count and overlay diameter ensure that the overlay scales well from a few to millions of peers.

5.2. Low delay

In the context of live streaming, peers benefit from a low distribution delay by adopting the proposed push-to-pull-based content distribution mechanism. Peers are enabled to choose among a number of neighbors to immediately forward incoming data blocks. Peers may favor close-by and non-congested peers. As a result, while the number of hops taken by data blocks grows logarithmically with the number of peers, delays remain almost

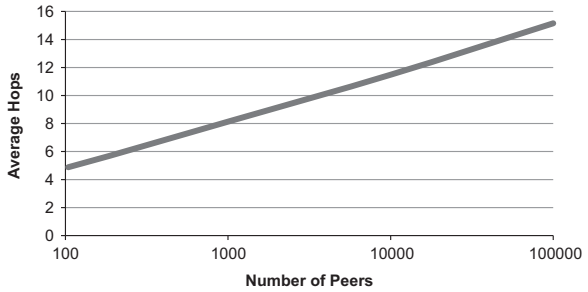

Fig. 6. Average number of hops taken by a block to reach a destination peer for 100 up to 100,000 peers.

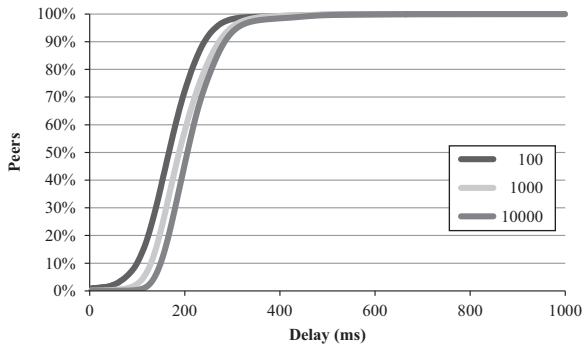

Fig. 7. Delay in milliseconds required for a data block to reach a given ratio of peers for different overlay sizes. The topology awareness allows the delay to remain almost constant regardless of the number of peers.

constant. For a larger number of peers, more close-by neighbors are available to forward blocks.

In the next scenario, the emulator makes use of a simulated network environment whereas peers are uniformly at random distributed within a bounded two-dimensional Euclidean space. The network latency between two peers is determined by the sum of the distance in the Euclidean space and an additional constant latency of 3 ms. The maximum latency, bounded by the size of the Euclidean space, is 200 ms, giving a maximum round trip time of 400 ms. This reflects the Internet latencies observed among distant locations, such as Europe and Australia. The used methodology is conceptually simple to focus on the proposed techniques and capture their essential properties. One may consider such a random distribution of peers in the Euclidean space a particularly bad distribution. In real world deployments, peers are likely to be clustered by continents, countries, regions, cities, or Internet service providers.

Fig. 7 depicts the distribution delays of the proposed protocol for different numbers of peers. For this scenario, the source is located at the center of the simulated topology and reachable by any peer within 100 ms, providing a lower bound for the achievable delay. On average, a data block reaches a peer after 170 ms for 100 peers. The average delay grows to 196 ms for 1000 peers and 214 ms for 10,000 peers. For the larger networks, a considerable part of the additional delay can be attributed to the 3 ms of local processing that the emulator enforces with each hop. In real world scenarios, the processing delay strongly depends on the capabilities and the network access of the deployed peers. Increasing the number of fixed bits β from 1 bit to 2 or more bits, i.e., increasing the number of children at each hop in the push phase, allows for a further reduction of the delay. The impact of congestion on the distribution delays is studied in a subsequent subsection.

5.3. Push vs pull

Fig. 8 compares three different strategies: a push-only strategy, a hybrid push-to-pull strategy, and a pull-only strategy. The push-only strategy applies the techniques from [10] to avoid cut-off branches. For the hybrid push-to-pull strategy, pushing is limited to about 20%. The pull-only strategy resembles the more commonly used unstructured peer-to-peer systems, but still benefits from the bounded overlay diameter and the flexible neighbor selection policy.

The push-only approach outperforms the other strategies in terms of delay. It reaches half of the peers after 160 ms and all peers after 220 ms. The push-only approach is able to maintain its performance lead as long as packet loss and malicious behavior of peers are negligible; otherwise, source coding may counter cut-off branches at the cost of duplicates, causing peers to obtain more than $k+\epsilon$ blocks.

In contrast to the push-only strategy, the pull-only strategy suffers from high latencies, as notifications and requests have to be sent back and forth. A major part of the time is spent reaching the first 10% of all peers. However, once a sufficient distribution of fresh blocks is reached, pull exchanges perform nearly as fast as push exchanges. Thus, the hybrid push-to-pull strategy incurs only a moderate delay over a push-only strategy; the push phase fuels the subsequent pull exchanges. With a hybrid push-to-pull approach, the system further gains in flexibility as noted in Section 4.4.

5.4. Robustness to congestion

In heterogeneous environments, some peers are likely congested and exhibit significantly higher network latencies.

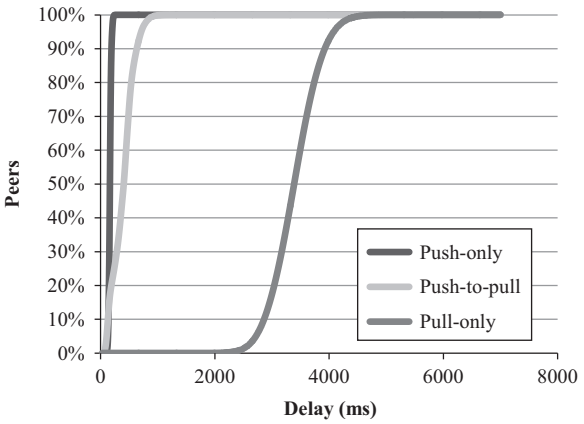

Fig. 8. Delay in milliseconds required by a data block to reach a given ratio of 10,000 emulated peers for different content distribution strategies.

The flexible nature of prefix-based overlays and the push phase enables peers to circumvent congested links. The push phase quickly attains sufficient distribution of new blocks among non-congested peers. The subsequent scenario considers 10,000 peers where 25% of the peers are congested. Like in the previous subsection, peers are uniformly at random distributed within a two-dimensional Euclidean area and the latency between peers is determined by their distance. Congested peers are assumed to have an additional network latency penalty of 200 ms. Without congestion, the average delay for a block to reach a peer from the source is 214 ms. With congestion, there is only a moderate additional delay of 10% for the non-congested peers. In contrast, the delay almost triples for the congested peers. The additional use of source coding can prevent additional delay for non-congested peers with high probability.

5.5. Temporal accuracy

Temporal scalability gives peers a more finely grained control of the download rate. While the push phase simply compares the temporal index of an incoming block with the advertised temporal indices of neighbors, the pull phase has to estimate the temporal index of a (missing) block based on the available blocks and their references to other blocks.

The following scenario studies a stream with seven temporal layers. The stream has a bitrate of 100 KB/s. Half of the peers have an unlimited download rate, while the others have a limited download rate between 10 and 100 KB/s. Push branches are cut off in order that 20% of the peers obtain a block in the push phase, while the others obtain it in the subsequent pull phase. Fig. 9 depicts the ratio of useful data downloaded by weak peers with respect to their download rate. Data is considered useful if it can be decoded by a scalable video decoder, i.e., all referenced data must be downloaded as well. The studied implementation performs well for practical purposes, with more optimizations possible. In most cases, peers are able to decode 90% or more of the downloaded data. For low bitrates, peers fail to decode a significant amount of data because there is insufficient capacity to download the temporal base layer (the key frames). In such a scenario, the use of additional spatial and quality layers is more useful. Note that the given example only considers the prioritization of data blocks in the base layer. Given a downloaded base layer, peers (exactly) determine the temporal indices of enhancement blocks by following the layer references. As a result, peers can make use of virtually all data downloaded from further enhancement layers.

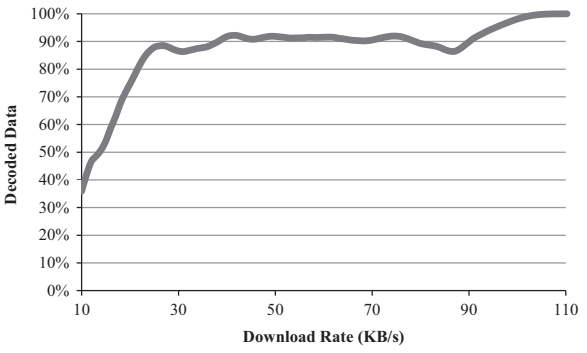

Fig. 9. Ratio of decodable data as a function of the maximal download rate for a base layer with seven temporal layers.

5.6. Fairness

The next scenario enforces a tit-for-tat-based exchange of content [11] to ensure fairness among peers. It applies the fairness mechanism to manage content exchanges among three different groups of peers. A first group has unlimited upload capacity, a second group has 110 KB/s upload capacity, and a third group has 50 KB/s upload capacity. The stream has a bitrate of 200 KB/s and partitions the data into three spatial and seven temporal layers. The example enforces a repayment ratio $\alpha = 1$ and a one-time credit $\gamma = 1$ applicable to a quarter of all blocks ($\delta = 4$). A repayment ratio $\alpha = 1$ forces neighboring peers to send (about) the same number of blocks to each other. The one-time credit allows a neighboring peer to download one block for free without having to give anything in return. However, to prevent large view exploits [12,26], the one-time credit only applies to a subset of all blocks ($1/\delta = 25\%$). Fig. 10 depicts the ratio of decodable blocks in relation to their spatial and temporal indices for all three groups. The resource-rich group obtains and decodes all blocks, while the weaker groups are limited to the lower two layers according to their upload capacities. For example, all peers in the weakest group manage to decode the lowest spatial layer, which has a bitrate that matches their upload limit. In accordance with the tit-for-tat model, these peers fail to obtain and decode most enhancement data.

5.7. Topology awareness

The emulated protocol favors the exchange of content among near-by peers to reduce the network load. Because of the use of a single overlay, peers come closer together and can more efficiently distribute base layers and lower enhancement layers among a larger number of peers. The result is a more efficient content distribution that can offset the overhead incurred by scalable coding, leaving only its benefits.

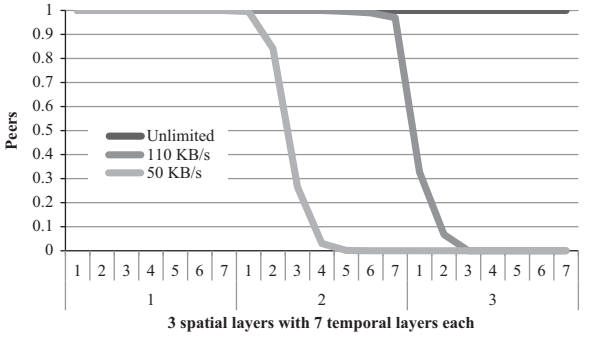

Fig. 10. Ratio of decoded blocks with respect to their spatial and temporal indices for three groups of peers with different upload capacities.

The following scenario considers an overlay with 30,000 emulated peers and three quality layers. Peers are again uniformly at random distributed within a two-dimensional Euclidean area. There is a base layer with two enhancement layers. The enhancement layers match the size of the lower layers, i.e., the base layer accounts for 25% of the bitrate, the first enhancement layer for 25%, and the second enhancement layer for 50%. In total, the third and best quality has four times the bitrate of the base quality. Peers are assigned uniformly at random to one of the three quality layers. The network load is defined as the sum of the size of all network packets weighted by the distance travelled. The total network load for three independent streams is 100%+200%+400%=700% the network load of 10,000 peers downloading the base quality (at a quarter of the bitrate). In contrast, if the peers join a single *scalable* overlay network, they benefit from additional near-by peers to exchange the base layer and the first enhancement layer. In our emulated protocol, an overlay with 20,000 instead of 10,000 peers incurs a 40% higher network load, while 30,000 instead of 10,000 peers incurs a 70% higher network load. The network load does not increase linearly because peers may obtain data blocks from near-by peers interested in different sets of layers that would otherwise belong to independent streams. Consequently, in the considered scenario, the network load grows to 170% for the base layer with 30,000 peers, to 140% for the first enhancement layer with 20,000 peers, and to 200% for the 10,000 peers in the second enhancement layer with twice the bitrate of the base layer. The total network load is 170%+140%+200%=510%, but an additional overhead of 10–15% for scalable coding has to be added. This corresponds to a reduction of 27% without the coding overhead and about 15% with the coding overhead. Additionally, scalable coding almost halves the traffic among distant peers. Only a single stream has to be distributed instead of multiple independent streams. In real world scenarios, this yields a lower Internet backbone load, which one may consider the most precious resource. The actual gain, however, strongly depends on the capabilities and locations of the participating peers.

5.8. Robustness to churn

The high connectivity of the proposed prefix-based overlay and the flexible choice of neighbors allow for building up and fixing routing tables quickly. As a consequence, a prefix-based overlay becomes easy to maintain and to recover even from massive concurrent changes. In contrast to most other peer-to-peer applications, it is essential to evaluate such scenarios in *live* streaming systems. With a live stream, all peers playback the same content at the same time and may react to this content. A commercial break may lead to a lot of people switching channels for a short period of time. The end of a program may trigger a large fraction of all peers to leave the channel. One may denote such behavior as *correlated* churn.

Four different kinds of events may occur: peers joining, peers leaving, peers switching coding layers, and peers switching playback positions in on-demand streams. For all four kinds of events, neighboring peers may have to update their routing tables accordingly. In this subsection, a scenario with 2000 peers and a 100 KB/s stream is considered. There are two coding layers: a base layer and a spatial enhancement layer of equal size. In the beginning, all peers download both layers. At some point, half of the peers switch to the base layer by setting their advertised spatial index from 1 to 0. In doing so, they no longer participate in the push phase of the enhancement layer and do not issue any pull requests for enhancement blocks. The (remaining) resource-rich peers downloading both layers have to repair their routing tables accordingly to still be able to forward enhancement blocks.

Neighbors are selected based on shared prefixes. For each shared prefix, peers maintain roughly two to three connections to other peers with identifiers that match the specific prefix. Each peer maintains the set of shared prefixes with their connections in a routing table. For both the switching peers and the (remaining) resource-rich peers, Fig. 11 depicts the number of accurate routing table entries and the download rates at a given time after half of the peers switched layers. The download rates are averaged over a period of 250 ms. As expected, the switching peers remain fully connected because any peer from either layer is suitable as neighbor. For approximately 70% of the stored prefixes at the resource-rich peers, at least one connection to another resource-rich peer is retained due to the backup connections. Only a second later, a suitable replacement is found for most entries. During this transition period, the resource-rich peers may still use the switching peers to forward enhancement blocks if no resource-rich peer is known. Moreover, the download rate of the switching peers declines rapidly allowing them to focus almost exclusively on the base layer. Given this fast repairing process, the protocol also copes well with any other kind of membership change.

Other kind of overlay structures have a harder time adapting in such scenarios. For example, Chunkyspread [27] uses a multi-tree structure and employs a variety of optimizations to ease construction and maintenance. If 10% out of 10,000 peers fail simultaneously, the average disconnect period for a peer is about 6 s, and the maximum disconnect period is 12 s. The introduction of additional

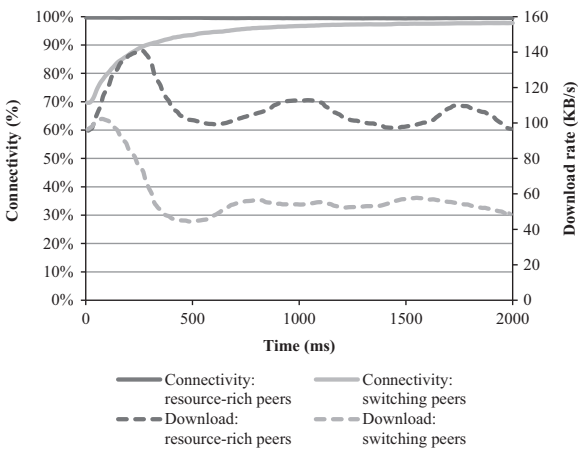

Fig. 11. Effect of half of the peers switching from the enhancement layer to the base layer. Shows the percentage of correct routing table entries and the download rate for both the switching and the remaining resource-rich peers. After 1 s, the peers are again almost fully connected.

redundancy is proposed by the adoption of source coding. A 18.75% redundancy (3 out of 16 slices) roughly halved disconnect durations. This comes at the cost of the same amount of additional traffic as peers usually obtain more than the minimum number of required blocks. From the perspective of a resource-rich peer that downloads all layers, this scenario is similar to the preceding scenario where peers switch layers. The proposed prefix-based structure is hardly affected and peers do not experience a disconnection period. Further emulations have shown that the results also hold if up to 90% of the peers are switching or leaving and if the overlay consists of up to 100,000 peers. Unstructured overlay networks also cope well in such scenarios, but fail to provide other important properties, like guaranteed connectivity, a low overlay diameter, and a low latency.

6. Conclusions

A variety of devices from mobile phones to televisions gained access to the Internet in recent years. For this purpose, we presented a peer-to-peer streaming protocol that is able to accommodate an arbitrary heterogeneous set of peers in a single overlay network. The protocol advocates an approach based on three complementing techniques: prefix-based overlays, scalable coding, and erasure coding. Prefix-based overlays make an efficient content distribution with low delay possible, guarantee connectivity, provide fairness, and have the flexibility to distribute scalable and erasure coded content. In doing so, scalable coding addresses the heterogeneity of peers by providing multiple coding layers that gradually improve playback quality. Erasure coding ensures that peers obtain the desired coding layers even if some of the blocks are lost due to congestion, churn, or malicious behavior. Moreover, the presented techniques are applicable to live, on-demand, and hybrid systems; easing the implementation of a system that supports multiple use cases.

References

[1] P. Baccichet, T. Schierl, T. Wiegand, B. Girod, Low-delay peer-to-peer streaming using scalable video coding, in: Proceedings of 18th Data Compression Conference (DCC), Snowbird, Utah, USA, 2008.

[2] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, SplitStream: high-bandwidth content distribution in a cooperative environment, in: Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS), Berkeley, California, USA, 2003.

[3] Y. Chu, S. Rao, H. Zhang, A case for end system multicast, in: Proceedings of International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), 2000.

[4] M. Feldman, K. Lai, I. Stoica, J. Chuang, Robust incentive techniques for peer-to-peer networks, in: Proceedings of ACM Electronic Commerce, 2004.

[5] V.K. Goyal, Multiple description coding: compression meets the network, IEEE Signal Processing Magazine (2001).

[6] G. Halkes, J. Pouwelse, Verifiable encryption for p2p block exchange, in: Proceedings of 10th IEEE International Conference on Peer-to-Peer Computing (P2P), Delft, The Netherlands, 2010.

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

[8] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in P2P networks, in: Proceedings of 12th International World Wide Web Conference (WWW), 2003.

[9] M. Karczewicz, R. Kurceren, The SP- and SI-frames design for H.264/AVC, in: Proceedings of IEEE Transactions on Circuits and Systems for Video Technology, 2003.

[10] T. Locher, R. Meier, S. Schmid, R. Wattenhofer, Push-to-pull peer-to-peer live streaming, in: Proceedings of 21st International Symposium on Distributed Computing (DISC), Lemesos, Cryprus, 2007.

[11] T. Locher, R. Meier, S. Schmid, R. Wattenhofer, Robust live media streaming in swarms, in: Proceedings of 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Williamsburg, Virginia, USA, 2009.

[12] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, Free riding in BitTorrent is cheap, in: Proceedings of Workshop on Hot Topics in Networks (HotNets), Irvine, California, USA, 2006.

[13] M. Luby, LT codes, in: Proceedings of 43rd IEEE Symposium on Foundations of Computer Science (FOCS), Vancouver, British Columbia, Canada, 2002.

[14] Z. Meng, T. Yun, Z. Li, L. Jian-Guang, Y. Shi-Qiang, Gridmedia: a multi-sender based peer-to-peer multicast system for video streaming, in: IEEE International Conference on Multimedia and Expo (ICME), 2005.

[15] S. Mirshokraie, M. Hefeeda, Peer-to-peer streaming with hierarchical network coding, in: Proceedings of IEEE International Conference on Multimedia and Expo, Beijing, China, 2007.

[16] S. Mirshokraie, M. Hefeeda, Live peer-to-peer streaming with scalable video coding and networking coding, in: Proceedings of 1st ACM Conference on Multimedia Systems (MMSys), Scottsdale, Arizona, USA, 2010.

[17] J. Noh, P. Baccichet, F. Hartung, A. Mavlankar, B. Girod, Stanford peer-to-peer multicast (sppm): overview and recent extensions, in: Proceedings of 27th Conference on Picture Coding Symposium (PCS), Chicago, Illinois, USA, 2009.

[18] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, A. Mohr, Chainsaw: eliminating trees from overlay multicast, in: Proceedings of 4th International Workshop on Peer-To-Peer Systems (IPTPS), 2005.

[19] C.G. Plaxton, R. Rajaraman, A. Richa, Accessing nearby copies of replicated objects in a distributed environment, in: Proceedings of 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), 1997.

[20] J. Pouwelse, J. Taal, R. Lagendijk, D. Epema, H. Sips, Real-time video delivery using peer-to-peer bartering networks and multiple description coding, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2004.

[21] I.S. Reed, G. Solomon, Polynomial codes over certain finite fields, SIAM Journal on Applied Mathematics (1960).

[22] A. Rowstron, P. Druschel, Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Proceedings of International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, 2001.

[23] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable H.264/MPEG4-AVC extension, in: Proceedings of IEEE International Conference on Image Processing (ICIP), Atlanta, Georgia, USA, 2006.

[24] K. Shin, D. Reeves, I. Rhee, Treat-before-trick: free-riding prevention for BitTorrent-like peer-to-peer networks, in: Proceedings of IEEE International Symposium on Parallel and Distributed Processing (IPDPS), Rome, Italy, 2009.

[25] A. Shokrollahi, Raptor codes, IEEE Transactions on Information Theory (2006).

[26] M. Sirivianos, J.H. Park, R. Chen, X. Yang, Free-riding in BitTorrent networks with the large view exploit, in: Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, Massachusetts, USA, 2007.

[27] V. Venkataraman, P. Francis, J. Calandrino, Chunkyspread: multi-tree unstructured peer-to-peer, in: Proceedings of 5th International Workshop on Peer-to-Peer Systems (IPTPS), Santa Barbara, California, USA, 2006.

[28] V. Vishnumurthy, S. Chandrakumar, E.G. Sirer, KARMA: a secure economic framework for P2P resource sharing, in: Proceedings of 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, 2003.

[29] W. Wang, D.A. Helder, S. Jamin, L. Zhang, Overlay optimizations for end-host multicast, in: Proceedings of Networked Group Communications, Boston, Massachusetts, USA, 2002.

[30] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, A. Luthra, Overview of the H.264/AVC video coding standard, in: Proceedings of IEEE Transactions on Circuits and Systems for Video Technology, 2003.

[31] X. Zhang, J. Liu, B. Li, Y. Yum, CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming, in: Proceedings of 24th IEEE Conference on Computer Communications (INFOCOM), Amsterdam, The Netherlands, 2005.