How to Adapt: SVC-based Quality Adaptation for Hybrid Peercasting Systems

Matthias Wichtlhuber*, Julius Rückert*, David Winter*, and David Hausheer*
* Peer-to-Peer Systems Engineering, Technische Universität Darmstadt, Germany
Email: {mwichtlh|rueckert|winter|hausheer}@ps.tu-darmstadt.de

***Abstract*—Live streaming of large-scale events such as the Olympic Games with a huge number of viewers is challenging, as the streaming infrastructure needs to scale fast and big, and often in an unpredictable manner. Peer-to-peer (P2P) live streaming (Peercasting) has proven to be beneficial in these scenarios, as resources are scaling inherently with the number of nodes. However, churn behavior in a node's neighborhood may result in fluctuating downstream bandwidth and thus freezing (stalling) playback. Related work tries to mitigate this effect by using layered video codecs, focusing on single-dimensional scalability in mesh-pull based systems. Yet, the benefits of multi-dimensional scalability (resolution, frame rate, and quantization) combined with coexisting pull-/push mechanisms introduced by modern hybrid P2P streaming architectures have not been studied in detail. Consequently, this work proposes a new scheduling algorithm taking these aspects into account. The evaluation shows large benefits for end-users by reducing the frequency of stalls by 90% even under extreme conditions.**

I. Introduction and Motivation

Delivery of video content is the predominant traffic source of today's Internet and is predicted to further increase in volume by several recent studies [3], [12]. This trend is rooted in the rapid proliferation of new devices and access technologies, e.g., tablets and smart TVs, as well as fibre and 4G connectivity. A large fraction of video traffic is caused by large Video on Demand (VoD) networks such as YouTube or Netflix and Content Delivery Networks (CDNs) such as Akamai. Besides VoD, there is also an increasing demand for streaming large-scale live events to a large audience, e.g., the Olympic Winter Games[1] in Sochi. These events have high requirements in terms of load and scalability on the underlying infrastructure. Moreover, it is necessary to fulfil these requirements in a short time frame, i.e., the infrastructure has to scale big and fast.

As IP multicast did not prevail in Internet Service Provider (ISP) networks, most streaming content is distributed via unicast today. Thus, bandwidth at the delivering Data Center (DC) or CDN increases linearly with the number of users. Especially services facing high (and/or unpredictable) traffic spikes can benefit from P2P approaches, because each client consuming the content also acts as a relay for other peers. Examples for P2P streaming deployments are the PPLive network [15], the recently released BitTorrent Live [4] system, and Akamai NetSession [19].

However, the advantages of P2P live streaming in terms of scalability and resource usage are often countered by the fluctuating bandwidth induced by peers joining and leaving the network (peer churn). As the transmitted video content usually has a more or less stable bit rate, fluctuation in bandwidth may lead to a degradation of quality by causing violated playback deadlines (stalling of the video), which is seriously affecting user's Quality of Experience (QoE) [5]. A more QoE friendly solution than letting the video stall is the adaptation of the video quality to the available bandwidth with layered video codecs (e.g., H.264/SVC [13]).

However, this approach induces interesting challenges for scheduling regarding the architecture of hybrid P2P live streaming systems. Hybrid P2P live streaming systems often employ an architecture using two delivery mechanisms in combination: a tree/push based delivery mechanism is used for stable peers, while pieces of data (chunks) not delivered via the tree are requested using a pull based mesh [18], [16]. As modern video codecs allow scaling the video quality along multiple dimensions (e.g., resolution, frame rate, and frame compression [13]), an effective scheduling algorithm does not only have to deal with push and pull based scheduling, but also has to decide simultaneously, which video quality along which dimension is to be delivered using which mechanism.

To address these challenges, this paper proposes a novel scheduling algorithm to cope with multiple delivery mechanisms of hybrid P2P streaming architectures as well as with layer selection along multiple dimensions at the same time. The proposed algorithm applies multiple layer quality filtering phases, accounting for the available bandwidth and video quality in a peer's neighborhood as well as user preferences.

The remainder of this paper is organized as follows: Section II provides background information on the topic. Section III describes the scheduler's design, which is evaluated in Section IV. Finally, related approaches are discussed in Section V, followed by the conclusion and outlook on future work in Section VI.

II. Background

The proposed scheduling algorithm utilizes a hybrid live streaming overlay and a layered video codec as outlined in the following.

A. Hybrid Overlay

The scheduler design presented is based on the TRANSIT [16] P2P live streaming overlay. The basic system architecture of the overlay is depicted in Figure 1. TRANSIT implements a hybrid tree-over-mesh approach, in which a *Flow Manager* manages delivery via a multi tree topology using one

[1] http://www.internetphenomena.com/2014/02/sochi-streaming-canadian-mens-hockey/, last visited 02.05.2014

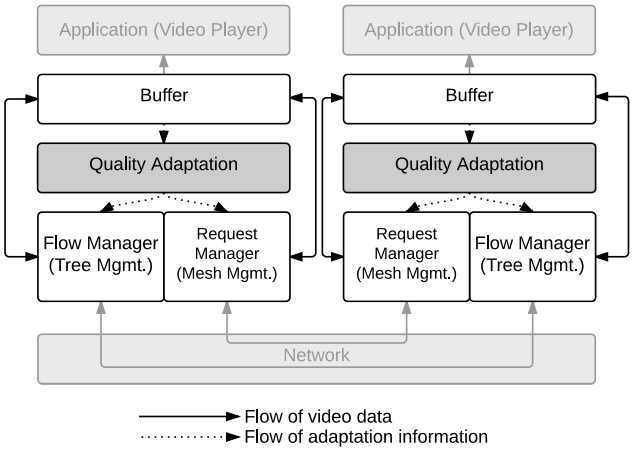


Fig. 1: Overview on streaming architecture.

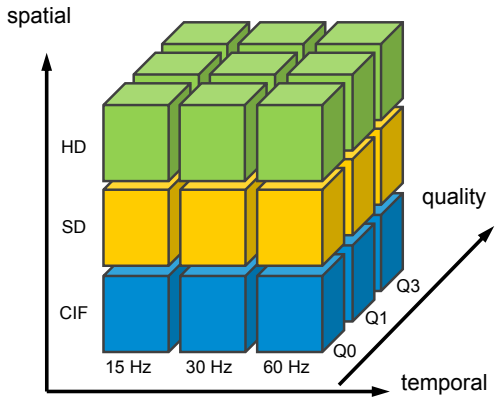


Fig. 2: SVC cube model with 3 different spatial, temporal and quality settings (reproduced from [1]).

tree per video layer and a *Request Manager* manages delivery via a mesh topology. The *Flow Manager* is an instance of a push based scheduling approach. Therefore, an agreement (*flow*) between a sender and a receiver is negotiated. The sender sends any chunk that was negotiated to be delivered to the receiver in silent consent. If chunk delivery fails (e.g., due to the sender leaving the network ungracefully), the contract is renegotiated with another party. As chunk delivery using flows relies on stable peer relations, the *Request Manager* handles all chunks that cannot be delivered via flows. Therefore, the *Request Manager* allows peers requesting and downloading single chunks of data from the overlay network based on frequently exchanged information on possessed chunks (*Buffer Maps*), thus serving as an instance of a pull based scheduler. This paper adds the building block of *Quality Adaptation* to the system, which decides based on information from the network and the buffer, which data is to be scheduled for download at which point in time. For additional details, see [16].

B. The Scalable Video Codec Extension of H.264

Most modern video transmission systems are based on fixed bitrate video codecs. These codecs do not allow implementing quality adaptation, as the video can only be played back if the entire data of the video is downloaded. Opposed to that, layered video codecs allow splitting a video into multiple layers. Layering the content allows to encode multiple quality versions into a single stream instead of encoding a separate stream for every quality layer. In particular, the Scalable Video Coding (SVC) extension of the H.264 standard [13] is used in this work, as the codec is standardized and mature encoders and decoders exist.

The H.264/SVC codec specifies scalability along three dimensions: frame resolution (*spatial*), frame rate (*temporal*) and frame quantization (*quality*). Figure 2 illustrates an example for a layered video that has been encoded with three different layers along each dimension. Since each layer is built on top of its predecessors to reduce redundancy, the so-called base layer ($(s,t,q) = (0,0,0)$), i.e., the smallest layer along all dimensions, is essential for video playback. If more layers are available, the received video quality is increasing at the expense of bandwidth required to download the data. This structure implies that layers are not independent of each other: if a layer $(s,t,q) = (2,2,3)$ is to be played back, all layers $\{(s,t,q)|s < 2, t < 2, q < 3)\}$ have to be present as well.

The possibility to change the layer during playback and adapting it to the current available bandwidth without opening new connections or wasting bandwidth by buffering other video representations make H.264/SVC a good candidate for P2P live streaming systems.

III. Scheduler Design

Before the proposed scheduler's rationale and design is described in detail, the definitions and terminology used are clarified. As we will frequently compare layers, we define a number of operators ($<'$, $>'$, $\leq'$, $\geq'$) to do so. We give a formal definition for the $<'$ operator here; the other operators are defined and used accordingly.

A layer (s_1, t_1, q_1) is defined to be smaller ($<'$) than another layer (s_2, t_2, q_2), if the following condition holds:

$$(s_1, t_1, q_1) <' (s_2, t_2, q_2) := \\ s_1 < s_2 \wedge t_1 < t_2 \wedge q_1 < q_2$$

Moreover, we define the set $L_{<'(s,t,q)}$ to contain all possible layers smaller ($<'$) than (s,t,q). We implicitly assume, that there is no layer (s,t,q), for which $(s,t,q) <' (0,0,0)$ holds true, i.e., the smallest possible layer is the base layer.

For the proposed scheduler design, the adaptation phase terminology of [1] is adopted and extended. Namely, the scheduling algorithm is divided into three phases: the *Initial Quality Adaptation (IQA)* phase, the *Progressive Quality Adaptation (PQA)* phase, and the *Final Layer Selection (FLS)* phase.

- The IQA determines an initial layer selection by setting an upper bound for the highest SVC layer. Therefore, the SVC cube is annotated with the marginal bandwidth requirements applied by the respective layer. Candidates are excluded, if the required bit rate is too high to receive the layer given the theoretical capabilities of the current interface. Moreover, layers could additionally be filtered by the resolution of the local screen or the computational

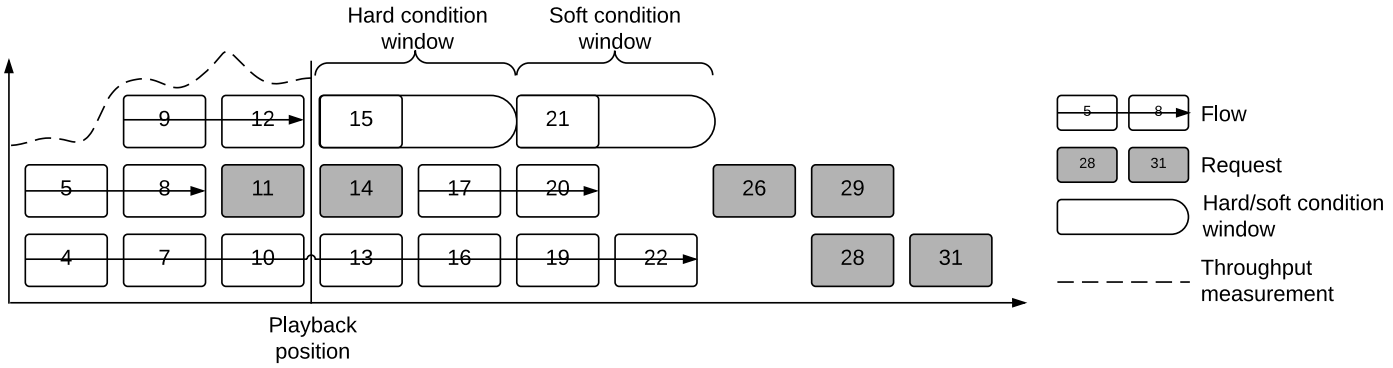


Fig. 3: SVC video buffer model (simplified, depicting a single dimension of the video).

capabilities of the device. As this process is trivial and only has to be executed once before playback starts, it is not discussed in detail in this work. Instead, a working IQA is assumed to exist, delivering a maximum layer $(s,t,q)_{IQA}$ suitable for the device and the theoretical maximum throughput of the device's interfaces.

- The PQA is introduced to adapt layer selection during runtime. Instead of the static local resources, real time information is extracted from the network and used to limit the range of allowed layers dynamically. However, this adaptation is done on the pre-filtered output of the IQA. Thus, PQA will never exceed the layer boundaries determined by IQA, i.e., $(s,t,q)_{PQA} <' (s,t,q)_{IQA}$.
- The described IQA/PQA phases are filtering phases that do not necessarily yield a single layer as a result. Thus, in the FLS phase, an algorithm finally selects the layer that matches best the user's preferences. Possible options can be to prefer either higher resolutions, frame rates or better frame quality settings.

The output of the filtering phases is a layer (s,t,q) defining a maximum layer to be scheduled, that is passed to the *Flow Manager* and the *Request Manager*, which then try to negotiate the respective flows and request to download the respective chunks. In the following, the PQA phase and the FLS phase are described in detail under the presence of push- and pull-based delivery mechanisms.

Algorithm 1 REDUCEBYTHROUGHPUT(L)

Require: L as the set of layers to be reduced, where $L \subseteq L_{<'(s,t,q)_{IQA}}$.
Ensure: Set L does not violate the constraints set by the available bandwidth from the neighborhood.

```
for all (s,t,q) ∈ L do
    // Filter according to throughput from neighborhood.
    if SIZE+(s,t,q) > BANDWIDTH↓ then
        L ← L \ {(s,t,q)}
    end if
end for
// Return filtered layer set.
return L
```

A. Progressive Quality Adaptation

The playback quality, i.e., the layer that can be played back, is influenced by two environmental parameters during runtime. First, there is the *availability of download bandwidth* from the neighborhood of a peer. Second, the *availability of layers* in the neighborhood is crucial. These parameters are independent of each other: it might be possible for a peer to achieve a high theoretical download bandwidth, while not being able to download high layers because they are not available. At the same time, all layers might be available in the neighborhood, but the neighboring peers do not provide enough bandwidth to download these layers. Thus, a good scheduler has to take both parameters into account and has to adapt according to their variations. Moreover, the importance of layers has to be taken into account. Besides pure quality adaptation, freezing (stalling) of the video has to be prevented. Research in the area of Quality of Experience has shown, that a freezing video has a very severe effect on the QoE of end-users [5]. Thus, if playback is about to stall, the download of lower layers, especially the base layer, has to be preferred over higher layers.

Taking these considerations into account, the PQA is designed as depicted in Figure 3: First, the layers are excluded that exceed the current throughput received from the neighborhood. This reduction is handled by Algorithm 1, which requires an enumeration of all possible layer combinations as an input. Afterwards, each candidate layer (s,t,q) is tested. If the aggregated size

$$\text{SIZE}^{+}(s,t,q) = \sum_{\substack{(s',t',q')\in \\ L_{\leq'(s,t,q)}}} \text{SIZE(s',t',q')}$$

of all layers up to the layer in question exceeds the current download bandwidth received from the neighborhood ($\text{BANDWIDTH}_{\downarrow}$), it is removed from the set.

In order to account for the layer availability, layers are scheduled according to the number of future *Requests* in a streaming session, which is estimated from the playback buffer. Intuitively, *Requests* are used for filling missing layers ("holes in the buffer") which cannot be delivered using stable *Flows* from the neighborhood. As described in Algorithm 2, the number of future requests is estimated by doing a buffer look

ahead using two sliding windows over the BUFFERCOUNT primitive, which is defined as follows:

$$\text{BUFFERCOUNT}((s,t,q),[j;k]) = \sum_{i=j}^{k}(s,t,q)_i.$$

In this definition, $(s,t,q)_i$ is 1, if the i-th chunk in the buffer can be decoded and played back, i.e., all layers $(s',t',q')_i \leq' (s,t,q)$ were already downloaded, and 0 otherwise. Intuitively, BUFFERCOUNT counts the number of playable blocks in the buffer along a certain layer, taking into account, that even if data is present, it might be worthless if the layers below are missing. Notably, the definition ensures an immediate reaction if the base layer is missing. In that case, BUFFERCOUNT evaluates to 0 no matter which layers were already downloaded.

The BUFFERCOUNT primitive is used to define two sliding windows. The first sliding window is the *hard playback window*, spanning H future chunks after the current playback position p. Whenever a chunk is missing in this window, the layer is discarded immediately (line 3 in Algorithm 2). The hard playback window is complemented by the *soft playback window*. This window spans S future blocks starting at position $p+H$. If the amount of already downloaded blocks in the window for a certain layer is smaller than a threshold C, the layer is excluded from future downloads. Selecting too high values for H, S, C will reduce the video playback quality because the further the buffer is investigated, the less likely it will be filled with data and the lower will be the selected layer. On the opposite side selecting these parameters too small will lead to more layer switches because the algorithm may select a too high layer that is not available in the near future. In such a case, the algorithm is forced to switch back to a lower layer, frequently.

So far, the algorithm takes into account network throughput from the neighborhood and layer availability, but does not implement the preference for lower layers as discussed previously. *Flows* can account for the reliable delivery of important layers, as the concept is designed to be used for stable peers in the neighborhood. Thus, *Flows* are used to download with a preference for lower layers, which leads to the *Flow Manager* filling up the buffer from bottom to top by trying to negotiate *Flows* for the respective layers. This strategy mitigates the effects of missing blocks obstructing playback in the lower/base layer(s). Moreover, the waste of bandwidth is limited by the bottom-up strategy, as situations are prevented, where a peer downloads a high layer without possessing all smaller layers.

Algorithm 2 REDUCEBYAVAILABILITY(L,H,S,C)

Require: L as the set of layers to be reduced, where $L \subseteq L_{<'(s,t,q)_{IQA}}$, H as the size of the hard playback window, S as the size of the soft playback window, C as the max. number of missing blocks in the soft playback window.
Ensure: Set L does not violate the hard and soft playback window condition.
1: **for all** $(s,t,q) \in L$ **do**
2: // Check hard condition, p refers to playback position.
3: **if** BUFFERCOUNT$((s,t,q),[p;p+H]) < H$ **then**
4: $L \leftarrow L \setminus \{(s,t,q)\}$
5: **end if**
6: // Check soft condition, p refers to playback position.
7: **if** BUFFERCOUNT$((s,t,q),[p+H;p+H+S]) < C$ **then**
8: $L \leftarrow L \setminus \{(s,t,q)\}$
9: **end if**
10: **end for**
11: // Return filtered layer set.
12: **return** L

Algorithm 3 REDUCEBYWEIGHT$(L, w[][])$

Require: L as the set of layers to be reduced, where $L \subseteq L_{<'(s,t,q)_{IQA}}$, a two dimensional array $w[][]$ containing adaptation weights.
Ensure: Final layer l has the highest weight of all layers $\in L$.
1: $l \leftarrow (0,0,0)$
2: **for all** $(s,t,q) \in L$ **do**
3: // Calculate weight.
4: **if** WEIGHT$((s,t,q),w) >$ WEIGHT$((s',t'.q'),w)$ **then**
5: $l \leftarrow (s,t,q)$
6: **end if**
7: **end for**
8: **return** l

B. Final Layer Selection

Since the PQA is designed as a filter running along all three dimensions of the SVC video at the same time, there might still be more than one selectable layer after PQA is applied. The FLS algorithm determines the final layer to be played back. Therefore, possible transitions between layers along each dimension of the SVC cube are annotated with weights to express the preferences of the user in terms of layer adaptation along the respective dimension (see Figure 4). As the weights are set per dimension, they can be stored easily in a two dimensional array data structure:

$$w[][] = \{\{w_{s1}, w_{s2}, \ldots\}, \{w_{t1}, w_{t2}, \ldots\}, \{w_{q1}, w_{q2}, \ldots\}\ldots\}$$

Following the example array depicted in Figure 4, a user has a preference to switch to a higher temporal layer from

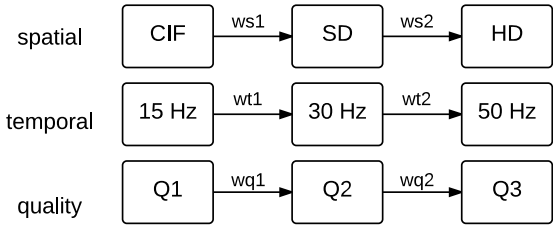


Fig. 4: Layer transition weight model with three spatial, temporal and quality levels.

the base layer first, and prefers a higher frame quality from there on. Setting the weights array allows to specify compact profiles for different device types, e.g. mobile devices. The source server can distribute a number of profiles together with the other streaming meta data once before streaming starts, where the peer picks a profile suiting the device's properties. After selecting the appropriate weight array, the FLS calculates a weight for each layer using the WEIGHT function:

$$\text{WEIGHT}((s,t,q),w) = \sum_{i=0}^{s-1}(w[0][i]) + \sum_{j=0}^{t-1}(w[1][j]) + \sum_{k=0}^{q-1}(w[2][k])$$

Whenever more than one layer is left by the PQA filter, the FLS algorithm selects the layer for which the WEIGHT function returns the highest value (see Algorithm 3).

C. Flow Probing

The scheduler described so far only reduces layers, but does not provide a way to scale up the video. To provide this feature, the capabilities of the neighborhood have to be probed, as the upper limit with respect to throughput and layer availability cannot be known in advance. For that purpose, again the inherent properties of *Flows* are utilized by increasing the maximum layer after IQA, PQA, and FLS filtering by one layer in one dimension following the gradient of the WEIGHT function (*Flow Probing*). If the probed flow can be negotiated with a peer from the neighborhood, this indicates two things: the delivering peer has enough spare capacities to serve the layer and has an own incoming flow for the respective layer, thus justifying an increase of the maximum allowed layer to be scheduled.

Note that increasing the layer increases the measured bandwidth capacity ($\text{BANDWIDTH}_{\downarrow}$) used by Algorithm 1, thus closing the quality adaptation feedback loop.

IV. EVALUATION

The proposed scheduler is implemented on top of TRANSIT as outlined in Section II-A, while the evaluation is conducted making use of the event-based PeerFactSim.KOM [14] simulation framework[2].

A. Evaluation Setup

The *bandwidth distribution model* of the peers in the system is based on the annual bandwidth penetration measurements of the OECD [9] (see Table I for details). In order to apply stress to the system, the upload and download bandwidths of high bandwidth peers were divided by two, which provokes a sufficient bandwidth bottleneck across the network to make frequent layer re-scheduling necessary: while the overall available download bandwidth stays constant, the overall upload bandwidth is cut down by 40%. The system is fed by two source servers providing 30 MBit/s upload bandwidth each. For connections between peers, a statistical *latency model* with a normally distributed delay of 100 ms and a variance of 50 ms is used.

The assignment of bit rates to SVC layers is based on bit rate measurements of an SVC encoded video. The video scales along the spatial and temporal dimension (see Table II), as the quality dimension was found to have minor impact on the visual quality while not influencing the bit rate per layer heavily. The layer scheduling parameterization was optimized in a separate parameter sensitivity study by varying one parameter at a time. The look ahead values $H = 3$, $S = 7$, and $C = 4$ were found to be optimal across all layers.

The overlay was optimized towards the workload in a separate parameter sensitivity study by varying one parameter at a time independently of the other parameters. As an outcome, peers serve at most 32 incoming and 32 outgoing connections per peer, where a connection may stay inactive for at most 15 seconds, before it is replaced by a new one. Moreover, buffer maps are exchanged every 2 seconds to inform neighbors on possessed video data.

To model churn adequately, Peers join and leave the system based on a measurement trace[3] of the PPLive streaming system measured by Vu et al. [15]. The trace contains more than 37'000 distinct hosts, which can hardly be simulated even when using modern hardware. Thus, the workload is scaled down to 200 present peers, preserving the joining and leaving behavior and the typical diurnal pattern. The number of 200 peers is chosen as it allows for moderate simulation duration while allowing to achieve significant results, as shown in Section IV-C. When joining the system, each peer draws an upload/download bandwidth and an IQA layer from the distribution shown in Table I. One hour of the trace containing a flash-crowd like joining behaviour is simulated.

B. Evaluation Metrics and Modes

For the evaluation of the scheduling algorithm, four metrics covering the temporal and quality aspects of the video playback are defined:

- The *Stalls Count* measures the average number of stalling events per minute, where a lower count indicates better performance, as fewer layer switches indicate a more stable performance of the scheduler. If stalling occurs, not only the frequency of stalling events is of interest, but also the average *Stalls Duration* in seconds.

[2]https://sites.google.com/site/peerfactsimkom/, last visited 3.8.2014.

[3]http://p2pta.ewi.tudelft.nl/datasets/t1407, last visited 29.4.2014

		temporal layer (t)			
		0	**1**	**2**	**3**
spatial layer (d)	**0**	731	1100	1483	1616
	1	1603	2445	3425	3878
	2	2772	4328	6378	7836
	3	4660	7296	10882	13957

TABLE II: SVC bitrates per layer in kBit/s.

Type	Down BW	Up BW	Max. Peer Count	$(s,t,q)_{IQA}$
Low	15300	2250	112 (56%)	(1,1,0)
Mid	42000	3150	60 (30%)	(2,3,0)
High	96400	52670	28 (14%)	(3,2,0)
High (stress)	48200	26335		
Server	-	30000	2	-

TABLE I: Peer bandwidth specifications in kBit/s and node count per bandwidth class [9], rows marked by (stress) are reduced values for applying stress to the system.

- The *Playback Percentage* measures the ratio of the time a peer spent playing back the video (t_p) compared to the time span the peer was present in the system (t_s). It is defined as:

$$PP = \frac{t_p}{t_s}$$

  Ideally, this metric is near 100% indicating permanent playback and no stalling.

- The *Relative Received Video Quality* metric measures the received video quality by comparing the bandwidth of the received layers to the bandwidth of the layer defined by the IQA (see Tables I/II). It is defined as:

$$RRVQ = \frac{\text{SIZE}^{+}((s,t,q))}{\text{SIZE}^{+}((s,t,q)_{IQA})}$$

  If a peer is able to receive a value of 100%, the peer was able to permanently download the selected IQA layer.

- The *Layer Switch Count* measures the number of layer switches. A small number of layer changes is desirable, as frequent changes can have a negative impact on the QoE [20].

For evaluating the scheduler design, three different PQA modes are compared representing a varying range of the aggressiveness of layer adaptation:

- *Disabled*: The layer scheduling is disabled for the entire experiment. If the scheduler is unable to receive the IQA layer, it is stalling. This mode serves as a base line for comparison.

- *Enabled*: The adaptation is invoked with a low frequency (every ten chunks, which translates to an invocation every 2.5 seconds). This mode emulates a prediction of future stalls with a low precision.

- *Schedule on Stall*: The layer scheduling is invoked with a high frequency (every chunk). Consequently, this mode reacts immediately, whenever a stall is about to happen.

C. Evaluation Results

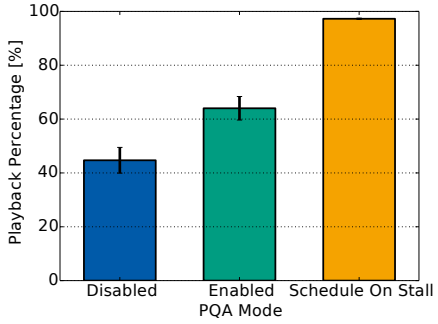
The evaluation results for the different modes and metrics are based on 5 runs of experiments. All results are averaged over 60 seconds of playback time over all peers and are reported with 95% confidence intervals, if not stated otherwise. As the evaluation setup contains heterogeneous groups of nodes with respect to bandwidth capabilities and layer selection, not only the average performance is studied, but also the corresponding cumulative probability functions (CDFs) are discussed to investigate the metrics' distribution among peers.

The results clearly reflect the benefit of the scheduler's design for the measured metrics. First, the stall related metrics are discussed as shown in Figure 5. While modes other than the *Disabled/Enabled* mode provoke frequent stalling, the *Schedule on Stall* mode reaches more than 95% *Playback Percentage* on average (see Figure 5a), which is an improvement of 60 percentage points over the *Disabled* mode. The stability of the result is reflected in the comparably small confidence interval and the corresponding CDF depicted in Figure 5b. The latter shows a fraction larger than 95% of all peers to reach a minimum of 95% *Playback Percentage* in the *Schedule on Stall* mode, while the other modes can reach a comparable performance for less than 20% in *Enabled* mode and 10% in *Disabled* mode only.

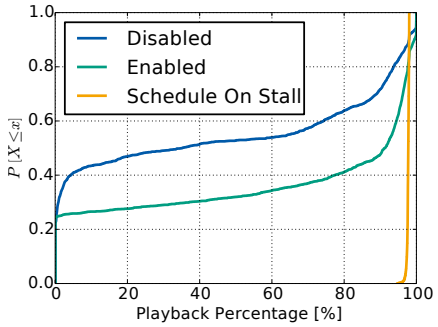
The *Playback Percentage* as a metric does not reflect the frequency and length of stalling events directly. Thus, the *Stalls Count* and *Stalls Duration* have to be studied as well. Figure 5c shows the CDF of the *Stalls Duration* metric for all three modes. Notably, the CDF shows that the *Schedule on Stall* mode is able to limit the maximum stalling time to less than two seconds, whereas the *Enabled/Disabled* mode can only guarantee 20 seconds and 48 seconds respectively. The CDF also shows the number of peers not facing stalling events at all, which is reflected by the y-intercept. In *Schedule on Stall* mode, this fraction of stalling free peers is as high as 80%, while in *Enabled* and *Disabled* mode only half as many peers can enjoy stalling free playback. The average *Stalls Count* is depicted in Figure 5d. The metric decreases from 0.25 stalls per minute to 0.025 stalls per minute, which is an improvement of roughly 90% at an increased stability as indicated by the confidence interval.

However, these improvements come at the cost of a lower video quality. Figure 6a depicts the video quality related metrics which will be discussed in the following. The *Relative Received Video Quality* drops for about 20 percentage points

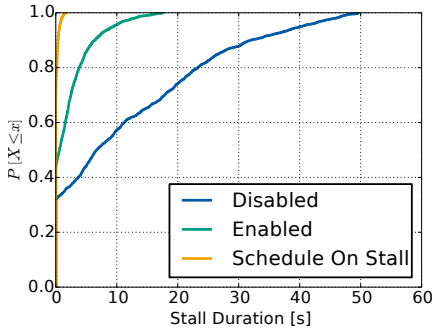
when comparing the *Disabled* and the *Schedule on Stall* mode, while the *Enabled* mode's performance is in between. This performance is expected and caused by the adaptation of the quality by the PQA/FLS. At the same time, the confidence intervals indicate that the performance is not deviating largely from the average, which shows that PQA/FLS do not induce layer switches over a large span of layers, thus keeping the quality deviation small during playback. The corresponding CDF depicted in Figure 6b confirms this result by showing a lower *Relative Received Video Quality* in *Schedule on Stall* mode than in *Enabled* mode. Nevertheless, both modes can achieve a *Relative Received Video Quality* near 100% for more than 50% of all peers. The *Disabled* mode is not depicted in this figure, as it always achieves 100 % *Relative Received Video Quality* at the cost of frequent stalling. It is reasonable to assume, that the frequency of layer switches has an impact on the perceived Quality of Experience. Thus, the effect of the scheduling strategy on the number of layer switches was studied in Figure 6c. As expected, the *Disabled* mode does not cause any layer switches. Generally speaking, the number of layer switches is below 0.3 switches per minute on average for all modes, i.e., on average there are more than three minutes between a layer switch. Surprisingly, the *Enabled* mode causes a higher number of layer switches than the *Schedule on Stall* mode, even though the *Schedule on Stall* mode is invoked at a higher frequency and should thus lead to a higher adaptation rate. This result indicates that the *Schedule on Stall* mode implements a more foresighted switching policy, performing a more careful evaluation of the bandwidth and layer availability conditions than the *Enabled* mode.



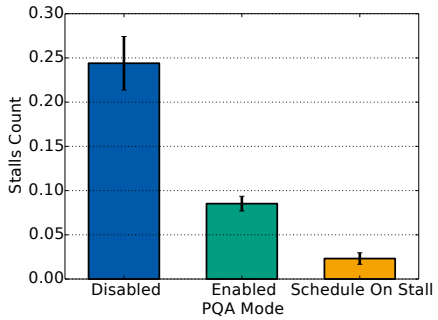
(a) Playback Percentage



(b) CDF of Playback Percentage



(c) CDF of Stall Duration



(d) Stalls Count

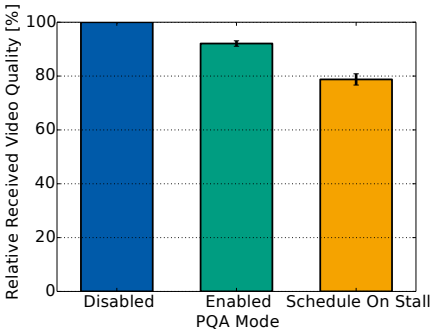
Fig. 5: Evaluation results of stall related metrics (5 runs, 95% confidence intervals, sampling interval 60s over all nodes).

Summing up, the stalling related metrics show a large performance benefit regarding the frequency and length of stalling events. Moreover, the amount of peers not stalling at all can be roughly doubled. Moreover, the evaluation of video quality related metrics revealed two general tradeoffs: first, stalling can be reduced in frequency and length by adapting the video quality. Second, the investigation of the *Layer Switch Count* showed that it is better to perform the adaptation with a high frequency, as fewer layer switches are caused at a comparably small drop in *Relative Received Video Quality*, but for a largely increased performance with respect to *Playback Percentage*.

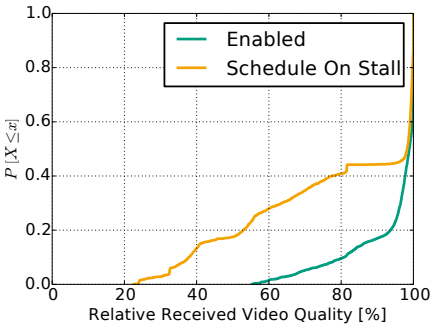
V. Related Work

The inherent dependencies between video layers of codecs, such as H.264/SVC, enabled the design of various centralized and decentralized adaptation mechanisms. The introduction of layered video codecs showed to be especially promising for P2P live streaming-based video streaming scenarios.

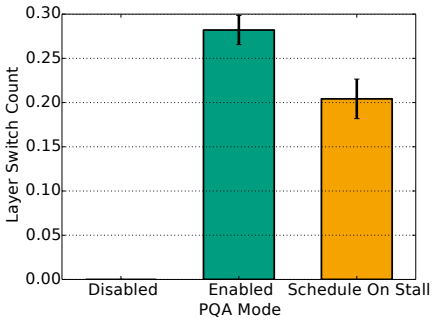
Regarding decentralized approaches in mesh-pull based VoD systems, the work presented in [8] provides an analytical model to predict throughput and quality of peers in SVC-based streaming systems. Complementing work by Abboud [1] shows how quality adaptation can be implemented for VoD streaming. The authors propose a two-stage adaptation mechanism for heterogeneous clients and show that stalling events can be reduced and even avoided by reducing the video quality at the clients in a controlled manner. The approach was further extended by [11] to also take different QoE aspects into consideration in the adaptation process. While these mechanisms were proposed for mesh-pull based VoD streaming scenarios, the work presented herein aims at live streaming using hybrid streaming topologies and push *and* pull based delivery mechanisms at the same time, which requires a significantly different algorithmic design. Moreover compared



(a) Relative Received Video Quality



(b) CDF of Relative Received Video Quality. *Disabled* mode not depicted as all nodes will gain 100% of quality in this scheduling mode.



(c) Layer Switch Count

Fig. 6: Evaluation results of video quality related metrics (5 runs, results averaged over $60s$ over all peers, 95% confidence intervals).

to [8], our work provides a realistic simulative evaluation in a carefully selected simulation setup based on validated models and user traces.

Other relevant related works such as the work by Bradai et al. [2] or Medjiah et al. [7] do investigate quality adaptation in a live streaming scenario, but do so using mesh-pull based systems. However, live streaming over mesh-pull based systems causes performance degradation, as the topology cannot account for the fast delivery of new content [18], [16]. Moreover, the work of Bradai et al. [2] focuses at keeping the numbers of video layer changes low and the amplitudes of adaptations small, following the observations by Zink et al. [20]. While this work addresses an important problem for SVC-based quality adaptations, it does not address the requirements of state-of-the-art hybrid streaming systems.

Moreover, different aspects of SVC-based quality adaptations have been studied in centralized, non-P2P live streaming contexts. Zhai et al. [17], for example, propose an SVC-based live streaming system for wireless, heterogeneous clients, aiming at maximizing the QoE for the end user. Kim et al. [6] go one step further and also move the decision for quality adaptations to a central component, basing adaptation decisions on estimations of the perceived quality. While these approaches show the potential of quality adaptations, in the work presented in this paper, the goal was to run the quality adaptations autonomously at the clients in a fully distributed scenario.

VI. Conclusions and Outlook

In this work, a novel scheduling algorithm for the distribution of scalable video content was proposed and evaluated. The proposed design is capable of handling SVC content scaling along multiple dimensions as well as chunk delivery over push and pull based mechanisms at the same time, thus fulfilling the needs of modern hybrid P2P live streaming architectures.

The scheduler is implemented atop of an existing hybrid streaming system [16] and evaluated in event-based simulation experiments. The simulation setup was chosen carefully and is based on validated models and user traces to be as realistic as possible. A set of metrics is defined reflecting the performance of the systems with respect to stalling events and video quality.

The evaluation revealed two trade-offs: the dynamic reduction of video quality can increase the performance of the stalling related metrics, i.e., the frequency and length of stalling events can be reduced to a large extent. The excellent performance (90% lower *Stalls Count*, 40 percentage points higher *Playback Percentage*) can be achieved even under stressful bandwidth conditions, i.e., a cut down of the overall available upload bandwidth by 40%. On the other hand quality adaptation comes at the cost of video quality. However, decreased video quality was found to be less influental on QoE than stalling [20]. Second, the frequency of the invocation of the quality adaptation has an impact on the number of quality adaptations. As opposed to the expectations of the authors, a more frequent invocation leads to less layer switches, as the quality adaptation acts in a more foresighted way.

Future work on this topic will focus on an even more concise integration of QoE aspects, e.g., by integrating visual impairment measurements between layers as they can be measured using the Video Quality Metric (VQM) [10], [11]. Impairment measurements can be integrated into the final layer selection as transition weights to perform a layer selection matching the user's perceived quality. Moreover, the scheduling approach presented in this work can be applied to different transport mechanisms with small modifications. Especially novel transport approaches like Dynamic Adaptive Streaming over HTTP (DASH) could be used as a replacement for flows and requests.

Acknowledgment

This work has been supported in parts by the European Union (FP7/#317846, SmartenIT and FP7/#318398, eCOUSIN) and the German DFG (CRC 1053, MAKI). We would like to thank Osama Abboud for fruitful discussions and Hongtao Zhang for careful editing.

REFERENCES

[1] O. Abboud, T. Zinner, K. Pussep *et al.*, "On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems," *ACM MMSys*, 2011.

[2] A. Bradai, U. Abbasi, R. Landa *et al.*, "An Efficient Playout Smoothing Mechanism for Layered Streaming in P2P Networks," *Peer-to-Peer Networking and Applications*, vol. 7, no. 2, pp. 101–117, 2014.

[3] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2012-2017," Tech. Rep., 2013.

[4] B. Cohen, "Peer-to-Peer Live Streaming," Patent US 0 066 969, 2013.

[5] T. Hoßfeld, D. Strohmeier, A. Raake *et al.*, "Pippi Longstocking Calculus for Temporal Stimuli Pattern on YouTube QoE," *ACM MoVid*, 2013.

[6] C. S. Kim, H. Sohn, W. D. Neve *et al.*, "An Objective Perceptual Quality-Based ADTE for Adapting Mobile SVC Video Content," *IEICE Information & Systems*, vol. E92-D, no. 1, pp. 93–96, 2009.

[7] S. Medjiah, T. Ahmed, and R. Boutaba, "Avoiding Quality Bottlenecks in P2P Adaptive Streaming," *Selected Areas in Communications*, vol. 32, no. 4, pp. 734–745, 2014.

[8] K. Mokhtarian and M. Hefeeda, "Analysis of Peer-Assisted Video-on-Demand Systems with Scalable Video Streams," *ACM MMSys*, 2010.

[9] Organisation for Economic Co-operation and Development, "OECD Broadband Report," Tech. Rep., 2012.

[10] M. H. Pinson and S. Wolf, "A New Standardized Method for Objectively Measuring Video Quality," *Broadcasting*, vol. 50, no. 3, pp. 312–322, 2004.

[11] J. Rückert, O. Abboud, T. Zinner *et al.*, "Quality Adaptation in P2P Video Streaming Based on Objective QoE Metrics," *IFIP NETWORKING*, 2012.

[12] Sandvine, "Spring 2013 Global Internet Phenomena Report," Tech. Rep., 2013.

[13] H. Schwarz and M. Wien, "The Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 135–141, 2008.

[14] D. Stingl, C. Gross, J. Rückert *et al.*, "PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems," *IEEE HPCS*, 2011.

[15] L. Vu, I. Gupta, K. Nahrstedt *et al.*, "Understanding Overlay Characteristics of a Large-Scale Peer-to-Peer IPTV System," *Multimedia Computing, Communications, and Applications*, vol. 6, no. 4, pp. 1–24, 2010.

[16] M. Wichtlhuber, J. Rückert, B. Richerzhagen, and D. Hausheer, "TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming," *IFIP NETWORKING*, 2014.

[17] G. Zhai, J. Cai, W. Lin *et al.*, "Three Dimensional Scalable Video Adaptation via User-End Perceptual Quality Assessment," *IEEE Broadcasting*, vol. 54, no. 3, pp. 719–727, 2008.

[18] X. Zhang and H. Hassanein, "A Survey of Peer-to-Peer Live Video Streaming Schemes – An Algorithmic Perspective," *Computer Networks*, vol. 56, no. 15, pp. 3548–3579, 2012.

[19] M. Zhao, P. Aditya, A. Chen *et al.*, "Peer-assisted Content Distribution in Akamai Netsession," *ACM IMC*, 2013.

[20] M. Zink, O. Künzel, J. Schmitt *et al.*, "Subjective Impression of Variations in Layer Encoded Videos," *IEEE/ACM IWQoS*, 2003.