TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming

Matthias Wichtlhuber*[1], Björn Richerzhagen†[1], Julius Rückert*[1] and David Hausheer*
* Peer-to-Peer Systems Engineering, Technische Universität Darmstadt, Germany
Email: {mwichtlh|rueckert|hausheer}@ps.tu-darmstadt.de
† Multimedia Communications Lab, Technische Universität Darmstadt, Germany
Email: richerzhagen@kom.tu-darmstadt.de

***Abstract*—The transmission of video content accounts for a large share of today's Internet traffic. While Video-on-Demand (VoD) substantially contributes to this, live streaming events such as video broadcasts from the Olympic Games can cause very high traffic volumes in the short term as well. Such peaks along with high fluctuations triggered by sudden changes in the behavior of users make the design of live streaming systems particularly challenging. Peer-to-Peer (P2P) has proven to be a scalable approach for disseminating content to a large number of users. Accordingly, the body of research offers numerous P2P live streaming approaches tailored towards specific scenarios and assumptions. However, no single approach is able to perform well under all possible conditions. Keeping up a high performance when conditions are changing is a challenge, since topology management and scheduling mechanisms cannot be exchanged easily. Therefore, this paper proposes TRANSIT, a new approach going beyond existing works in that it supports transitions between different live streaming mechanisms. TRANSIT makes different configurations of such mechanisms seamlessly exchangeable to enable the optimal choice of configurations for a wide range of live streaming scenarios. The approach is evaluated using measurement- and trace-based workloads. The results show that TRANSIT is able to maintain a high performance at a low overhead in highly fluctuating environments, whereas static configurations show serious performance degradations.**

I. INTRODUCTION

The delivery of video content is the predominant traffic source on today's Internet and forecasts imply a further increase of video traffic for the coming years [5], [18]. The reasons are manifold: consumer devices such as smartphones and tablets have been broadly adopted and are used increasingly to access content over the Internet [18]. Additionally, more and more TV sets and set-top-boxes allow easy access to online services.

Besides VoD services like YouTube and Netflix, the streaming of live events such as the Soccer World Cup, the Olympic Games, or the Red Bull Stratos event play an increasingly important role. The design and operation of live video streaming mechanisms for such events is extremely challenging, mainly due to the unpredictable behavior of users, inducing high dynamics that those mechanisms have to account for. Starting from a small number of users, a stream may suddenly become popular, requiring the streaming mechanisms to rapidly scale to millions of users. In [18] such effects are documented for the Olympic Games 2012, accounting in peak times for almost 22% of the overall Internet traffic in the US and showing a high fluctuation in the number of viewers over the entire games.

In order to deal with such peaks and fluctuations, commercial streaming providers today typically build on expensive large-scale server infrastructures, often in combination with content delivery networks (CDNs). At the same time, P2P streaming systems such as PPLive[2] and the recently released BitTorrent Live[3] have emerged as an alternative approach. By leveraging idle client resources (primarily bandwidth), those systems are able to scale well with the number of participating users, while reducing the costs for the streaming provider.

Existing P2P streaming systems can be decomposed into a number of basic building blocks (*mechanisms*, see Section II), which are optimized for specific scenarios and operating ranges. Those mechanisms typically perform well as long as the operating conditions do not change. However, no single mechanism exists which is able to maintain a high streaming performance over the entire range of operating conditions for a large-scale live event, while keeping the costs for the provider low. The reason for this is that in highly dynamic scenarios like large-scale live events, where conditions (e.g., in terms of number of viewers) can change rapidly, maintaining a high performance is very challenging, since topology management and scheduling mechanisms cannot be changed easily.

A. Goal and Contribution

To tackle those challenges, this paper proposes a new approach going beyond existing P2P live streaming approaches by enabling transitions between different mechanisms and configurations. A transition is defined as the switch from one mechanism configuration to another one in order to maintain or improve the streaming performance under changing operating conditions. In this context, a mechanism configuration can refer to a single mechanism implementation or a combination of different ones.

The concept of transitions is understood as a fundamental design principle as presented in [17]. It goes beyond the notion of traditional adaptive streaming mechanisms in that it allows for the integration and use of arbitrary streaming mechanism

[1]Authors contributed equally.

ISBN 978-3-901882-58-6

[2]http://www.pptv.com/ [Accessed October 7, 2013]
[3]http://live.bittorrent.com/ [Accessed October 20, 2013]

implementations. Therefore, two key mechanisms for P2P-based video streaming systems are examined in this paper: the *topology management mechanism* and the *data scheduling mechanism* (see Section III).

In order to study the impact of transitions under varying operating conditions, the sudden appearance of a large number of users (*flash crowd*) is the focus of this work. For addressing this specific aspect of large-scale live streaming scenarios, two hypotheses are defined:

1) A quantifiable difference between alternative streaming mechanism configurations exists during a flash-crowd scenario that justifies a transition between them.
2) Transitions between mechanism configurations allow for a high and stable streaming performance with low overhead and costs for the content provider.

To prove these hypotheses, this paper contributes TRANSIT, a highly flexible approach that on one hand can be used to extensively study quantitative trade-offs between different streaming mechanism configurations and, on the other hand, allows to systematically design transitions between these configurations. In particular, transitions of topology (*multi-tree* and *mesh-based* topology), and scheduling (*push-based* and *pull-based* scheduling) at runtime are investigated. These transitions are triggered in order to maintain a high performance under changing conditions faced by the system. To this end, TRANSIT provides a fully-fledged P2P live streaming system supporting transitions between different topology management and scheduling mechanisms. At the same time, the approach provides means to fix mechanisms to a certain state to study the impact of transitions as well as the benefits and drawbacks of individual mechanisms.

The remainder of this paper is structured as follows: Section II introduces streaming mechanisms that were identified in the related work and further motivates the concept of transitions. Based on this discussion, requirements for a transition-enabled streaming approach are derived and the design of TRANSIT is presented (Section III). Section IV evaluates the transition-enabled approach by comparing it against several fixed mechanism configurations in order to prove the constructed hypotheses. In Section V, conclusions are drawn and further directions for the application of the transition paradigm are discussed.

II. BACKGROUND AND RELATED WORK

Existing approaches in P2P video streaming can be classified according to the applied topology, which is usually either *single-tree-*, *multi-tree-*, *mesh-based*, or *hybrid*.

Single-tree-based systems form a virtual overlay tree originating from a streaming source towards all clients. The topology allows to perform a *push-based* delivery by initiating all packet transfers on the sender side. However, as these systems (e.g., OVERCAST [8]) have problems to cope with high churn, multi-tree-based systems were developed. Multi-tree-based systems often come in conjunction with a coding scheme such as Multiple Description Coding or Scalable Video Coding, which allow to split the stream into substreams that can be decoded independently to some degree (SPLITSTREAM [4], COOPNET [13], [14]). Each substream is fed into a subtree, where all peers participate in all subtrees, but at different positions to mitigate the effect of peer failures. In a mesh-based system (PRIME [10], CHAMELEON [11]), peers usually form a random mesh topology. Missing data is requested based on frequently exchanged information on available video data (*pull-based* scheduling). Due to a higher redundancy of connections, mesh-based systems are inherently more robust to churn than tree-based systems. The increased robustness comes at the price of higher overhead for the management of connections. In contrast to meshes, tree-based overlays are efficient in terms of overhead. However, peer churn has a higher impact on the performance of those systems [9], [20].

Hybrid systems try to find a good compromise between single topology management and scheduling mechanism configurations as discussed above. The authors of TOMO [3] propose a mesh-over-tree approach, where the overlay is multi-tree-based and stable peers form an additional mesh. The explicit assignment of peer types and topologies as proposed for this system seems counter-intuitive, as a tree-based approach exposes its strengths mostly in stable environments. The MTREEBONE [24] streaming system maintains a single-tree-based overlay based on stable peers to support data delivery with low latency. To increase robustness, a mesh overlay is formed to aid in maintenance and packet delivery. The primary method of scheduling is the push-based delivery via the single-tree, while pull-based requests are sent to the mesh as a fallback in case of packet loss. While MTREEBONE [24] forms a stable backbone by distilling reliable peers, LIVESKY [26] deploys a CDN to provide reliable delivery to the edges of the network. To lower the load, peers connected to the CDN additionally form a hybrid P2P system, using a multi-tree approach supported by a mesh for receiving packets that cannot be retrieved otherwise.

Hybrid systems contribute ideas to the design of a transition-enabled streaming system. However, the concept of transitions as applied in TRANSIT differs from the concept of hybrid overlays. Hybrid overlays aim to find a good compromise between a *fixed* number of mechanism configurations to optimize the delivery process for specific scenarios, whereas the idea of transitions is to freely *switch* between mechanism configurations in order to support an open range of scenarios and operating conditions. Such a transition could, for example, be the switch from a tree-based topology management mechanism as used by OVERCAST [8] to a mesh-based topology management mechanism resembling PRIME [10].

The choice of the scheduling and topology management mechanism determines the performance of a P2P streaming system under various operating conditions. For the above approaches, those mechanisms are fixed. TRANSIT, however, uses fine-grained combinations of push- and pull-based scheduling as well as topologies in between mesh and multi-tree, depending on the operating conditions. Moreover, if required by the conditions, TRANSIT is also able to apply a full-blown mesh or multi-tree topology.

III. TRANSITIONS IN P2P LIVE STREAMING

In order to investigate the benefits of transitions, TRANSIT has been designed with the following requirements in mind:

Transition of Topology and Scheduling Mechanisms: The system has to support transitions between pure mesh-based, hybrid, and pure multi-tree topology management mechanisms. Topology management transitions imply transitions of the scheduling mechanism. Therefore the system has to support different scheduling mechanisms such as pull, push and any hybrid combination.

Seamlessness: All of the aforementioned transitions are to be executed seamlessly, that is, the requirements for live streaming have to hold before, during, and after a transition.

Decentralized Control: Transitions are executed by peers without any central management and control. Therefore, peers have to evaluate their environment and initialize a transition based on their local knowledge. This requires the system to tolerate transitions to be executed by only a subset of peers, implying the concurrent application of several mechanisms.

System Convergence: The system has to converge to a stable state if the operating conditions do not change significantly. Particularly, oscillations, i.e., transitions at a high frequency caused by slight variations of the environment have to be avoided in order to keep the overhead low.

The discussed requirements are specific to transitions and affect the user only indirectly during playback. As the environment may change over time or even differ depending on a peer's location, mechanisms have to be switched during operation without affecting the overall Quality-of-Service (QoS). Therefore, besides the above transition-specific requirements, the following application-level requirements are to be fulfilled:

Low Startup Delay: A user wants to experience low startup delay when joining a stream. The startup delay is affected by the size of the buffer that needs to be filled before playback starts and the effectiveness of the joining process.

Continuous Playback: Once the stream starts playing, it should not be interrupted due to missing packets or an empty buffer. The number and length of such interruptions affects the overall perceived QoS.

A. TRANSIT *Design*

A multitude of mechanisms need to coexist and communicate with each other in a transition-enabled system. Due to the asynchronous nature of the system, different peers in the system can be in a different state of a transition. Transitions are not triggered by a central entity but are based on local knowledge of each individual peer. Thus, the state of a peer and its currently applied mechanisms might differ from the states of its neighbors. TRANSIT distinguishes between mechanisms on two functional layers, as shown in Figure 1. Mechanisms on the topology management layer ensure connectivity with other peers, while mechanisms on the scheduling layer ensure the delivery of the video stream. Mechanisms communicate with each other by using a unified protocol.

The concept of *connections* is used to provide a well-defined service interface between the scheduling layer and the topology management layer. Those connections are formed by the mechanisms on the topology management layer. Once a connection is established, it can be used by the scheduling layer for the transmission of video data. Unused connections are closed after a given time and replaced with new ones. Therefore, the scheduling layer and its utilization of connections directly determine the applied topology.

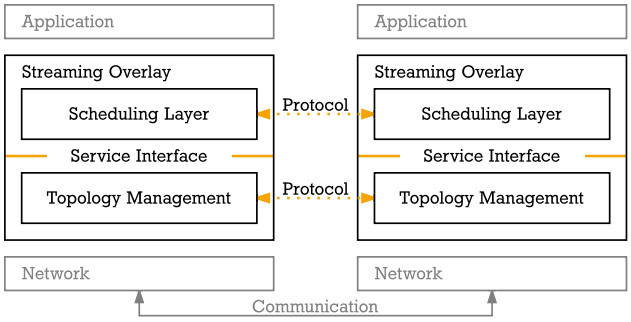


Fig. 1. System overview of TRANSIT.

TRANSIT's foremost goal is providing a constantly high QoS by varying mechanisms. The system converges towards a mesh-/pull-based system in highly fluctuating environments to maintain playback at higher overhead, whereas stable scenarios result in a multi-tree-/push-based configuration to prevent waste of bandwidth. This flexibility is achieved by introducing the concept of *flows* and *requests*, which allows to apply any scheduling approach in the range between push and pull.

Video blocks are either requested (pulled) or delivered (pushed). The latter assumes a contract between the sender and the receiver. If the contract is not fulfilled anymore, it has to be renegotiated, possibly with a new counterpart. In a multi-tree-based system, this is done by determining a new parent peer for the given stream. With increasing churn rates, the reliability of negotiated contracts declines. In an extreme case, the contract has to be renegotiated for each block of the transmission, resulting in a push-based system behaving like a pull-based one, as each block is requested separately.

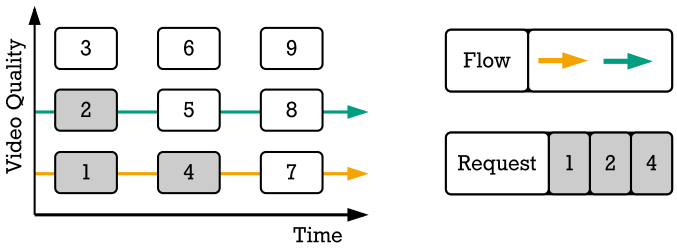
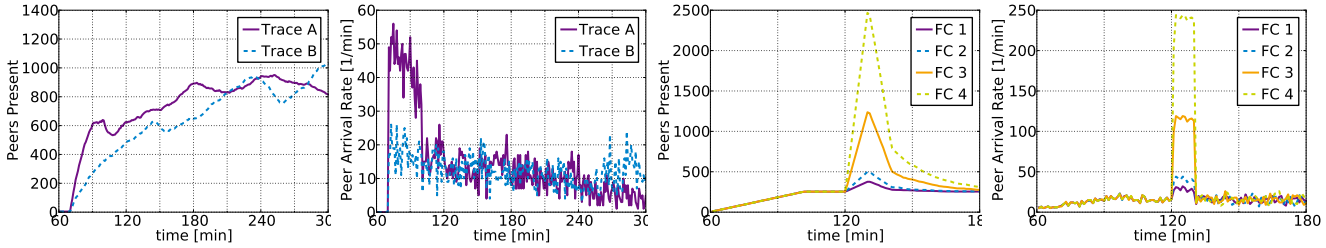


Fig. 2. Addressing scheme for flows and requests.

The main differences of flows and requests are illustrated in Figure 2. TRANSIT uses the Scalable Video Codec (SVC) [19] extension of the H.264/AVC [22] video standard, which was shown to be beneficial for supporting peers with heterogeneous resources in [1], [16]. SVC-encoded content can be decoded at different quality layers, where only parts of the full bit stream are needed to decode the video at a lower quality. Thus, the



(a) Number of peers (b) Arrival rate (c) Number of peers (d) Arrival rate

Fig. 3. Characteristics of the trace-based workload (Figures 3a and 3b) and the flash crowd workload (Figures 3c and 3d).

addressing scheme can be represented by a matrix, where the vertical direction stands for increasing quality layers and the horizontal direction shows the chronological order of chunks, i.e., pieces of the video, as they are rendered. The atomic unit in this scheme is a block, which represents a certain quality layer of the respective chunk. The addressing scheme for individual blocks is kept simple by assigning consecutive numbers to each block. A flow represents the durable delivery of the most recent blocks of X layers of a stream, e.g., blocks $\{1, 2, 4, 5, 7, 8, ...\}$ (equivalent to the first two layers of the stream). Requests represent the delivery of arbitrary blocks, such as blocks $\{1, 2, 4\}$.

The definition of flows and requests should not be mistaken for the categorization into push- and pull-based scheduling. As described above, a flow might ultimately lead to pure pull-based scheduling if it is very short-lived, whereas a request might easily contain hundreds of blocks that are delivered one after the other in a pure push-based fashion. By using the concept of flows and requests, arbitrary scheduling schemes can be modeled. This allows seamless adaptation of the scheduling mechanism to different operating conditions.

A lesson learnt from related work is the importance of fast transmission of new chunks [10], [24]. Consequently, their fast dissemination is defined as the first and foremost goal. It does not matter if some packets get lost on the way, if a request mechanism is provided that is able to retrieve missing packets in time. This further motivates the concept of flows and requests as defined above, with flows forwarding the most recent blocks of a stream, regardless of currently missing blocks in a peer's buffer or its playback position.

Consequently, TRANSIT has a preference for flows, requests are emitted in the case of failing flows only. As a newly negotiated flow delivers only the most recent packets, the missing packets have to be requested. This simple priority forces the topology into a mesh in highly fluctuating environments, whereas stable scenarios form a multi-tree topology. Further details about the design of TRANSIT are documented in [15].

IV. EVALUATION

The evaluation focuses on comparing the performance of different scheduling mechanisms and topology management mechanisms using trace-based workloads and workloads derived from real-world measurements in order to create a realistic setting.

This comparison can be done with TRANSIT, as it can be configured as a pure mesh or a pure multi-tree based system, as well as any combination of both. The goal of this evaluation is to assess the performance of the underlying mechanisms, for example pull- vs. push-based scheduling mechanisms. The evaluation motivates the support for transitions between mechanisms in a live streaming system by understanding when such a transition is beneficial for the overall system.

In order to evaluate the impact of transitions on a streaming system's performance, the configurations *mesh/pull*, *mesh with newest-first scheduling/pull*, *multi-tree/push*, and TRANSIT are compared under varying workloads. The evaluations were conducted using the simulation framework Peerfact-Sim.KOM [21].

A. Simulation Setup

The evaluation of TRANSIT uses two workload models, one based on measurement traces of PPLIVE and the other being a workload derived from measurements of flash-crowd scenarios.

1) Trace-based Workload: The traces used in this work were collected by Vu et al. [23] in a measurement study of PPLIVE and are provided in the P2P trace archive [27]. The characteristics of the traces closely resemble traces used in earlier studies by Hei et al. [6] and have also been the basis of a recent study by Hu et al. [7]. Relying on publicly available traces for the evaluation provides a good estimate of the system behavior under real-world conditions.

Figure 3a shows the number of concurrently online peers, while Figure 3b depicts the corresponding arrival rates. The traces differ in the arrival rate, especially during the startup phase where trace A exhibits an arrival rate of roughly 45 peers per minute. Later, both traces behave quite similar, with an arrival rate of around 15 peers per minute and between 600 and 1000 concurrently online peers.

2) Flash-Crowd Workload: In this model, peers join the system with an assigned session length until the threshold $|\tilde{P}|_{\max}$ of concurrently online peers is reached. The distribution of session lengths is derived from the aforementioned measurement trace [23]. After time t_{fc} the number of online peers is increased linearly up to $m_{\text{fc}} \cdot |\tilde{P}|_{\max}$. This increase happens during the *flash-crowd interval* d_{fc} which is fixed to ten minutes. By fixing the interval length and by varying the target number of peers, different arrival rates can be modeled.

Peers that join during this interval are not replaced with new peers if their session expires. Thus, the number of online peers converges back to $|\tilde{P}|_{\max}$ as more and more sessions end. Finally, if the number of online peers drops below $|\tilde{P}|_{\max}$, leaving peers are again replaced with new ones. The resulting characteristics of the workload for different flash-crowd intensities are shown in Figure 3c and Figure 3d.

The applied scenario consists of a massive join of peers during the flash-crowd phase and a massive leave shortly afterwards, as the session lengths are distributed exponentially. Both situations occur in real-life systems as well, for example on New Year's Eve [6] or after unforeseen events such as earthquakes [23], of course depending on the content the video stream is offering. The flash-crowd workload captures these extremes, thus providing a realistic scenario to evaluate the benefits of a transition-enabled approach in such a situation.

A peer is assigned to one of the groups specified in Table I based on the results described in [7], [12]. The group determines the peer's upload data rate $\hat{R}_{\uparrow}$ and download data rate $\hat{R}_{\downarrow}$ as well as the bit rate of the quality layer $rate(l)$ it is interested in. Access link data rates are based on a study conducted by the Organisation for Economic Co-operation and Development (OECD) [12]. The table furthermore shows the distribution of nodes to those groups according to the OECD study. Each peer is assigned an SVC quality layer based on its access link bandwidth. The data rates of those requested SVC layers are derived from [7]. As the focus of this work is not on quality adaptation, the requested layer does not change during playback.

TABLE I
ACCESS LINK DATA RATES, FREQUENCY OF PEERS AND REQUESTED VIDEO BIT RATES (DERIVED FROM [7], [12]).

Group	$\hat{R}_{\uparrow}$ (Mbit/s)	$\hat{R}_{\downarrow}$ (Mbit/s)	Perc.	$rate(l)$ (kbit/s)
DSL	2.25	15.3	56%	503.0
Cable	3.15	42.0	30%	705.3
FTTH	52.67	96.4	14%	905.8
Sources	5.0	5.0	3 Sources	–

B. Evaluation Metrics

The choice of metrics presented in this section is based on the work of Zhang et al. [28], who defined a set of widely applicable evaluation metrics for streaming systems.

Connection Frequency: The connection frequency f_C is defined as the number of incoming connections that were newly formed during a fixed *sampling interval* Δ, indicating how the topology management layer reacts to peer churn.

Protocol Overhead: Overhead is defined as the ratio between payload and additional protocol traffic. The overhead in terms of messages being sent is termed $O_{\#}$, and the overhead in transmitted bytes is termed O_s. Duplicate video packets are counted as part of the overhead.

Startup Delay: The startup delay d_{su} describes the time an individual peer has to wait after he joined the system, until the first chunk is played. It is determined by the time it takes a peer to connect to the system and to receive enough chunks for the player to start playback. This, in turn, is affected by the buffer size that is configured in the system.

Playback Smoothness: The playback smoothness describes how often a video stream of a single peer is interrupted due to missing chunks and how long these interruptions last. It is derived from [2], where the authors present metrics to assess a peer's session quality. With the session length of a peer denoted by δ and duration (s) returning the length of a stall $s \in S$, with S being the set of all stalls of the peer, the *playback smoothness* Ψ is defined as

$$\Psi = 1 - \frac{1}{\delta - d_{\text{su}}} \cdot \sum_{s \in S} \text{duration}\,(s) \quad . \tag{1}$$

Playback Performance: As the startup delay is not taken into account when calculating the playback smoothness, a second metric is defined that qualifies the whole session of a peer in terms of video playback. Therefore, the *playback performance* Υ of a peer is calculated as

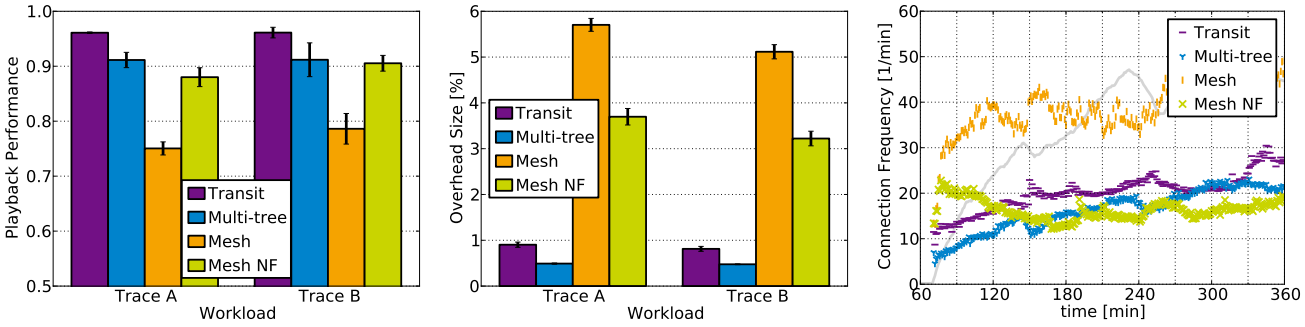
$$\Upsilon = 1 - \frac{1}{\delta} \cdot \left(d_{\text{su}} + \sum_{s \in S} \text{duration}\,(s) \right) \quad . \tag{2}$$

C. Evaluation of Transitions

In the following, all experiments are repeated five times with different seeds for the random generator. Results are reported with 95 % confidence intervals, if not stated otherwise.

1) Behavior under Trace-based Workload: The trace-based workload model is used to emulate the different configurations under real-world conditions. The performance of individual mechanisms does not change significantly over time, as the conditions given by the workload are rather stable. Figure 4a shows the aggregated playback performance for both traces and all systems, and Figure 4b shows the overhead that is necessary to reach this performance. TRANSIT reaches the best performance of more than 95 % at an overhead of below 1 % of the total data being sent. The multi-tree configuration performs comparably well and requires even less data overhead than TRANSIT. Considering the mesh-based configurations, the newest-first selection strategy exhibits superior performance over the earliest-deadline-first strategy, resulting in an increase of roughly 15 % in playback performance. Due to the more efficient scheduling process, connections are not changed as frequently, leading to a decrease in data overhead.

The connection frequency over time is presented in Figure 4c, showing a significantly lower connection frequency for the newest-first mesh when compared to the other mesh configuration. The solid line indicates the number of concurrently active peers in the system due to the workload. The connection frequency for the multi-tree-based configuration follows this line in that the frequency increases with the number of online peers. The average connection frequency of TRANSIT rises slightly during phases where peers leave the system. This is an indicator of the efforts of the topology management layer to maintain a set of connections for the negotiation of flows and requests. Due to the newest-first



(a) Playback performance Υ (b) Overhead size O_s (c) Connection frequency over time

Fig. 4. Performance and cost of the evaluated mechanisms for the trace-based workload.

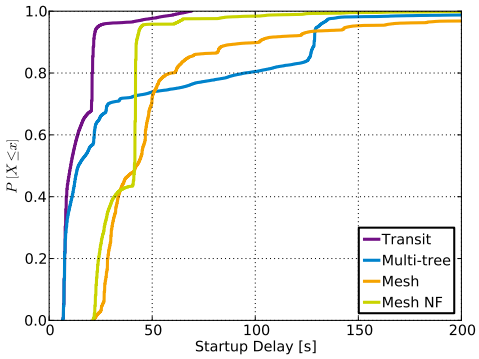


Fig. 5. Startup delay distribution in trace-based workload.

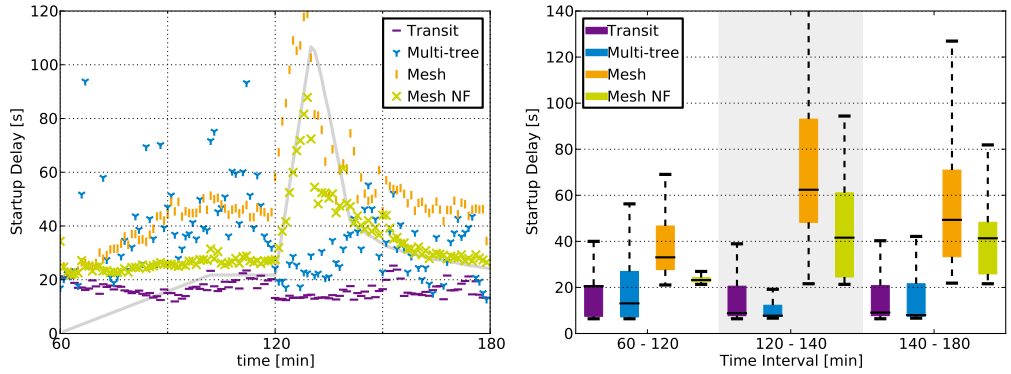
selection strategy, the respective mesh profits from nearly every connection. Connections are not closed as long as they are used for the transmission of payload, leading to low connection frequencies. In contrast to that, the pure mesh configuration changes neighborhoods frequently, as especially peers that just joined the system have little chance of finding a neighbor that can provide the newest blocks. The importance of the packet scheduling strategy in a live streaming system becomes clearly visible from these results.

The distribution of startup delays as shown in Figure 5 shows the advantage of TRANSIT over the mesh-based solutions, as TRANSIT is able to achieve a much lower startup delay. The multi-tree performs comparably well, but has problems with a fraction of the peers that request the video at a higher quality layer. Those peers need significant amounts of time to find subtrees for each of their requested layers, leading to high startup delays.

2) Behavior under Flash-Crowd Workload: For analyzing each of the defined metrics, scatter plots of time series are used for the flash-crowd workload. Each of these plots is complemented by a whisker plot showing the performance of the configuration under test in the three main phases of the workload. The intervals are chosen as follows: *startup phase* (60-120 min), *flash-crowd phase* (120-140 min), and *cool down phase* (140-180 min). In these plots, the boxes represent the first quartile q_1 and the third quartile q_3, while the whiskers cover all values smaller than $q_3 + 1.5(q_3 - q_1)$ or greater than $q_1 - 1.5(q_3 - q_1)$ of *all* samples of *all* simulation runs.

Figure 6a (6b) shows the startup delay and Figure 7a (7b) the playback smoothness at different times throughout the experiment. Each depicted data point is an aggregation of all values collected over one minute. The startup delay is plotted in the interval in which the respective peer joined the system, while the playback smoothness is averaged in each interval over all currently online peers.

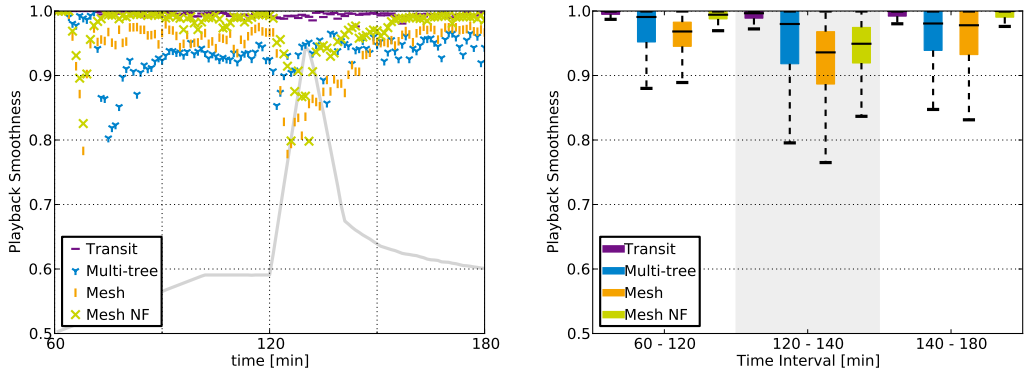
The high arrival rate of new peers during the flash-crowd phase has severe impact on the startup delay in both mesh configurations. This is due to the fact that with an increasing number of peers more hops are needed to distribute new packets from the source throughout the system. Therefore, with each intermediate peer, the overall delay increases. As new peers start their playback at the most recent chunk in the system, they have to wait until those new packets reach one of their neighbors. While the newest-first approach helps in reducing this effect, it is still outperformed by TRANSIT and its fast content dissemination through flows. The startup delay as well as the playback smoothness in the multi-tree configuration show high variation before and after the flash-crowd phase, i.e., during a time where peers arrive at low rates. During the flash-crowd phase, the multi-tree configuration shows rather stable values and outperforms both mesh-based approaches. This seems counterintuitive at first, given that mesh-based systems are generally considered to be more robust against churn than multi-tree-based systems [9]. However, this effect is due to two important properties of a live streaming system. First, all peers are interested in roughly the same chunks of the video stream at the same time. As already mentioned, the mesh suffers due to the increased number of requests needed to distribute a new packet, even if newest-first is used as scheduling strategy. The multi-tree is able to distribute new packets very fast at least to a subset of all peers, as long as parts of the subtrees remain intact. Second, the chance of retrieving a missing block increases with the number of neighbors. Each time a peer in the multi-tree configuration leaves the system, missing flows on its children have to be re-established and missing blocks have to be requested from neighbors. During the flash crowd, the performance of the request mechanism of the multi-tree configuration increases, as the connection frequency and thereby the number of potential sources for missing packets, increases as well. This is



(a) Startup delay d_{su} over time

(b) Startup delay d_{su} within workload phases

Fig. 6. Startup delay for the individual mechanisms during the flash-crowd scenario over time. The whisker plot shows aggregated information during the startup phase, flash-crowd phase, and cool down phase. The mesh-based mechanisms experience increased startup delay during the flash-crowd phase.



(a) Playback smoothness Ψ over time

(b) Playback smoothness Ψ within workload phases

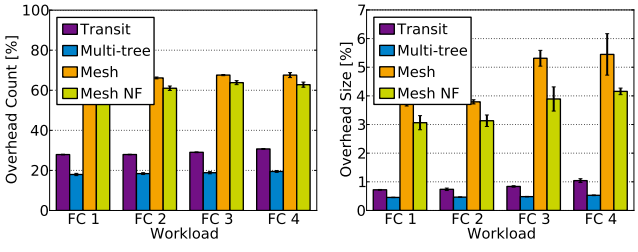
Fig. 7. Playback smoothness for the individual mechanisms over time. The whisker plot shows aggregated information during the startup phase, flash-crowd phase, and cool down phase. The multi-tree mechanism distributes the most recent blocks faster during the flash-crowd phase, leading to a better playback smoothness compared to the mesh-based mechanisms.

observable in Figure 8a (8b), where the connection frequency of all systems shows a peak during the flash-crowd phase.

One could argue that the mesh should perform better as well, given that it consists solely of a request mechanism that should profit from the increased connection frequency. Nevertheless, compared to the multi-tree configuration, the mesh is not able to distribute new packets fast enough to a sufficiently large fraction of peers to support the high number of new arrivals. This results in higher startup delays as well as decreased playback smoothness of both mesh approaches during the flash-crowd phase.

TRANSIT is able to maintain both, high playback smoothness and low startup delay, even during the flash-crowd phase. Figure 8a (8b) shows that the connection frequency of TRANSIT follows the pattern of the multi-tree configuration, with a much higher increase during the massive join phase. The data indicates TRANSIT to be able to maintain ideal properties for the request mechanism: a high number of neighbors and frequent changes of those neighbors. At the same time, the request mechanism benefits from the flow mechanism used in TRANSIT, as new packets are distributed to a large fraction of all peers very fast.

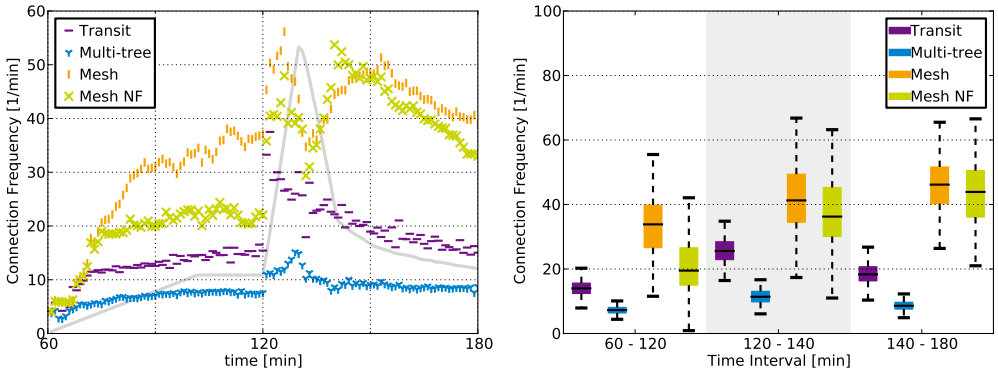
This becomes even clearer when expressing playback smoothness and startup delay as *playback performance* (Figures 9a, 9b). Each marker in the plot shows the aggregated playback performance for all peers that joined during the respective one minute phase. Therefore, a marker during the flash crowd is the average over a large number of peers that joined during this phase, whereas the markers before and after the flash-crowd phase stand for less peers. As shown in Figure 9a (9b), the mesh with newest-first request strategy exhibits a very high and, more important, *consistent* playback



(a) Overhead count $O_{\#}$

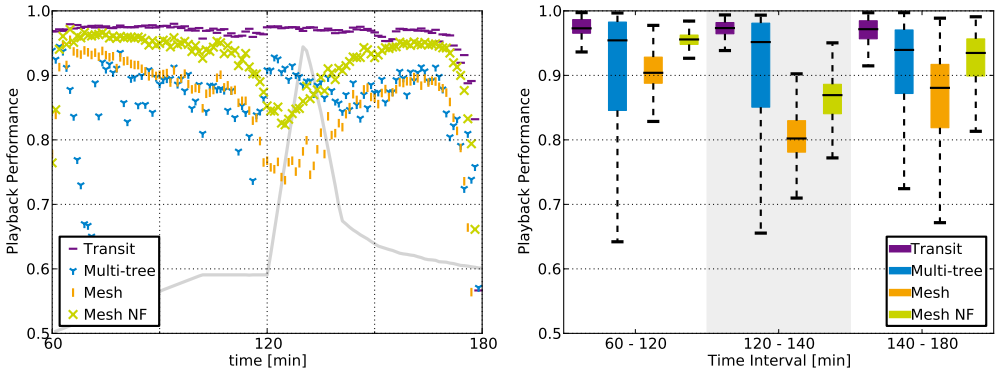
(b) Overhead size O_s

Fig. 10. Overhead during the flash-crowd scenario.



(a) Connection frequency f_C (b) Connection frequency f_C within workload phases

Fig. 8. Connection frequency indicating the frequency of neighborhood changes. During the flash-crowd phase, the mesh-based mechanisms aggressively establish new connections to maintain fluid playback and to distribute the newest video chunks.



(a) Playback performance Υ (b) Playback Performance Υ within workload phases

Fig. 9. Combination of startup delay and playback smoothness into a single metric, highlighting the trade-off between tree-based and mesh-based mechanisms during the flash-crowd phase.

performance for peers that joined before and after the flash crowd. The multi-tree does not perform much worse, but its performance is far more dependent on the initial neighborhood selection and the failures of other peers, leading to high variations. However, during the flash-crowd phase, the conditions for the multi-tree benefit the playback smoothness and the startup delay, resulting in superior and consistent performance for peers that join during this phase. TRANSIT combines the advantages of both mechanisms and achieves consistently high performance, regardless of the time a peer joins the system. The sharp decrease in playback performance towards the end of the experiment is due to the fact that the peers left the system after three hours. Peers that join during this last phase of the experiment experience a startup delay as illustrated in Figure 6a (6b), but their session time is much shorter, resulting in shorter playback times. As the playback performance combines startup delay and playback phase weighted with their respective lengths, the startup delay has a high impact on the calculation on these peers. The results of the individual metrics reveal those peers to exhibit the same high performance.

Figure 10 shows the overhead in terms of sent messages as well and actual bytes transmitted on average as a fraction of data being sent. Both mesh configurations show significantly higher overhead due to the pull-based packet scheduling. The overhead of all configurations increases slightly with the intensity of the flash crowd, indicating additional control traffic being exchanged due to connection maintenance. When considering the overhead in terms of actual data being transmitted, this effect becomes far more significant for the mesh-based configurations. Here, overhead accounts for roughly $4\,\%$ of all transmitted data, whereas it stays below one percent for TRANSIT and the multi-tree-based configuration. Still, with only a fourth of the overhead introduced by the mesh-based configuration, TRANSIT is able to provide superior playback performance. Additional evaluation results under additional workloads can be found in [15].

V. CONCLUSIONS AND FUTURE WORK

Based on related work, basic topologies and scheduling mechanisms have been identified as being influential on the performance of P2P live streaming systems. As most of the existing approaches are using these mechanisms in a fixed way and are thus tailored towards specific assumptions and scenarios [20] [9], the following hypotheses were constructed:

1) A quantifiable difference between alternative streaming mechanism configurations exists during a flash-crowd scenario that justifies a transition between them.

2) Transitions between mechanism configurations allow for a high and stable streaming performance with low overhead and costs for the content provider.

In order to confirm these hypotheses, this paper presented TRANSIT, a transition-enabled P2P streaming approach allowing for the systematic evaluation and comparison of different mechanisms for scheduling and neighborhood selection. TRANSIT does not maintain a fixed topology or scheduling mechanism configuration. Instead, the system has the ability to perform fine-granular transitions between a mesh and multi-tree topology as well as push- and pull-based scheduling mechanism configurations according to current environmental parameters (e.g. flash crowds). The system's high adaptivity is enabled by a clean separation of topology management and scheduling mechanisms and the concept of flows and requests.

TRANSIT's ability to resemble different system properties was used to compare the performance of fixed mechanism configurations and the transition-enabled system. Where a fixed mechanism configuration such as the pure mesh configuration had obvious problems, optimizations such as newest-first scheduling were introduced for a fair comparison.

The evaluation results indicate that TRANSIT offers very good resilience in flash-crowd scenarios. The important user-centric performance metrics such as startup delay and playback smoothness are nearly unaffected under massive churn, whereas mesh and multi-tree configurations show serious service degradation. TRANSIT reaches these results at the cost of comparably low overhead: as expected, the system acts slightly worse than the multi-tree configuration, but much better than both mesh configurations. This, in turn, confirms both hypotheses. Moreover, the presented results show that the overall concept of transitions is beneficial with respect to a broader range of potential mechanism configurations.

Further investigations are planned to improve existing mechanism implementations (e.g. by introducing more sophisticated scheduling modes to reduce start up delay) and for including additional mechanisms in the concept. A promising direction for new mechanisms is the area of network coding. Network coding techniques allow to save bandwidth at the cost of computational overhead by aggregating messages as a combination of the initial packets. By making the parameters of the combination and the aggregated message known at the receiver side, the packets can be decoded [25]. In future work, it will be investigated how network coding can be used as a further mechanism to allow for transitions to such schemes when bandwidth gets scarce.

ACKNOWLEDGMENT

This work has been supported in parts by the European Union (FP7/#317846, SmartenIT and FP7/#318398, eCousin) and the German DFG (Collaborative Research Center 1053, MAKI). The authors would also like to acknowledge valuable input and feedback by their colleagues and project partners.

REFERENCES

[1] O. Abboud, K. Pussep, A. Kovacevic *et al.*, "Quality Adaptive Peer-to-Peer Streaming Using Scalable Video Coding," in *IFIP/IEEE MMNS*, 2009.
[2] O. Abboud, T. Zinner, K. Pussep *et al.*, "On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems," in *ACM MMSys*, 2011.
[3] S. Awiphan *et al.*, "ToMo: A Two-Layer Mesh/Tree Structure for Live Streaming in P2P Overlay Network," in *IEEE CCNC*, 2010.
[4] M. Castro *et al.*, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *ACM SIGOPS*, 2003.
[5] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2010–2015," Tech. Rep., 2011.
[6] X. Hei, C. Liang, J. Liang *et al.*, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.
[7] H. Hu, Y. Guo, and Y. Liu, "Peer-to-peer Streaming of Layered Video: Efficiency, Fairness and Incentive," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 8, pp. 1013–1026, 2011.
[8] J. Jannotti, D. K. Gifford, K. L. Johnson *et al.*, "Overcast: Reliable Multicasting with an Overlay Network," in *ACM OSDI*, 2000.
[9] N. Magharei *et al.*, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," in *IEEE INFOCOM*, 2007.
[10] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1052–1065, 2009.
[11] A. T. Nguyen, B. Li, and F. Eliassen, "Chameleon: Adaptive Peer-to-Peer Streaming with Network Coding," in *IEEE INFOCOM*, 2010.
[12] Organisation for Economic Co-operation and Development, "OECD Broadband Report," Tech. Rep., 2012.
[13] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient Peer-to-Peer Streaming," in *IEEE ICNP*, 2003.
[14] V. N. Padmanabhan, H. J. Wang, P. A. Chou *et al.*, "Distributing Streaming Media Content Using Cooperative Networking," in *ACM NOSSDAV*, 2002.
[15] B. Richerzhagen, "Supporting Transitions in Peer-to-Peer Video Streaming," November 2012, Master's Thesis, Peer-to-Peer Systems Engineering, TU Darmstadt, Germany. [Online]. Available: http://www.ps.tu-darmstadt.de/fileadmin/publications/Ric12.pdf
[16] J. Rückert, O. Abboud, T. Zinner *et al.*, "Quality Adaptation in P2P Video Streaming Based on Objective QoE Metrics," in *IFIP NETWORKING*, 2012.
[17] J. Rückert and D. Hausheer, "Bridging the Gap: Towards an Adaptive Video Streaming Approach Supporting Transitions," in *AIMS*, 2012.
[18] Sandvine, "Fall 2012 Global Internet Phenomena Report," Tech. Rep., 2012.
[19] 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.
[20] Z. Shen, J. Luo, R. Zimmermann *et al.*, "Peer-to-Peer Media Streaming: Insights and New Developments," *Proceedings of the IEEE*, vol. 99, no. 12, pp. 2089–2109, 2011.
[21] D. Stingl, C. Gross, J. Rückert *et al.*, "PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems," in *IEEE HPCS*, 2011.
[22] I. T. Union, "ITU-T H.264 - Advanced Video Coding for Generic Audiovisual Services," Tech. Rep., 2010.
[23] L. Vu, I. Gupta, K. Nahrstedt *et al.*, "Understanding Overlay Characteristics of a Large-Scale Peer-to-Peer IPTV System," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 6, no. 4, pp. 1–24, 2010.
[24] F. Wang *et al.*, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast," in *IEEE ICDCS*, 2007.
[25] J. Xu, X. Wang, J. Zhao, and A. O. Lim, "I-Swifter: Improving Chunked Network Coding for Peer-to-Peer Content Distribution," *Peer-to-Peer Networking and Applications*, vol. 5, no. 6, pp. 30–39, 2011.
[26] H. Yin, X. Liu, T. Zhan *et al.*, "Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky," in *ACM Multimedia*, 2009.
[27] B. Zhang *et al.*, "The Peer-to-Peer Trace Archive: Design and Comparative Trace Analysis," in *ACM CoNEXT Student Workshop*, 2010.
[28] X. Zhang *et al.*, "A Survey of Peer-to-Peer Live Video Streaming Schemes - An Algorithmic Perspective," *Computer Networks*, vol. 56, no. 15, pp. 3548–3579, 2012.