Clubbing with the Peers: A Measurement Study of BitTorrent Live

Julius Rückert, Tamara Knierim, and David Hausheer
Peer-to-Peer Systems Engineering Lab, Technische Universität Darmstadt, Germany
Email: {rueckert|tknierim|hausheer}@ps.tu-darmstadt.de

***Abstract*—The peer-to-peer approach can greatly help to cope with highly dynamic live streaming workload by using idle client resources. Yet, P2P streaming typically comes at the cost of increased streaming delays caused by the inevitable multi-hop forwarding of content by peers within the overlay. Various P2P streaming approaches have been proposed aiming at a good tradeoff between flexibility, streaming delay, and costs in terms of traffic overhead for both content providers and clients. Recently, BitTorrent Inc. released a new P2P live streaming system termed BTLive, specifically targeted at low delay and low overhead. For content providers investigating the applicability of BTLive's approach, it is essential to understand its properties as well as its limitations. So far, no publicly available study exists that quantitatively analyzes BTLive's performance. To this end, this paper presents a measurement study of the official beta version of BTLive. The study aims to answer the following key questions: How peer-to-peer is BTLive? How delay optimized is BTLive? What is the overhead of BTLive? To answer these questions, traces of real BTLive traffic between a broadcast server and a number of peers deployed across Europe have been analyzed.**

I. Introduction

The distribution of media content over the Internet has gained increasing attention over the last decades, resulting in a dominating part of worldwide network traffic [3], [17]. The increasing access bandwidth of consumers combined with the development of highly efficient video codecs, as well as new classes of end-user devices are inevitably changing the user behavior in media consumption [11] and have coined new application scenarios and business models for over-the-top video streaming. Examples include video-on-demand services like YouTube[2] for user generated content or Netflix[3] for movies and TV series. Even traditional TV broadcasters recently started providing their content as catch-up TV. In spite of these new services, studies show that linearly broadcasted TV content as well as live delivery of events still play an important role for the users [10], [11]. As a result, also more and more live content shifts to the Internet [7].

The distribution of live video content on an Internet scale has been extensively studied both in research and industry. While IP multicast would be a desirable and efficient approach for distributing live streams to a large number of users, it has a number of considerable drawbacks for network providers that could not completely be addressed in the last decades [8]. As a result, IP multicast is not available on an Internet scale and, in particular, not for the delivery of over-the-top traffic. Today, mostly cost-intensive centralized streaming systems are used, relying on individual IP unicast streams to clients. Due to the inherent limitations of this approach, content-delivery networks (CDNs) were deployed, such as the largest one by Akamai [16]. With hundred thousands of servers all over the world, Akamai forms an overlay network on top of the Internet that is able to widely distribute content before it is delivered to the end users by nearby CDN nodes relying on unicast.

To further improve the distribution of content and to reduce costs for the content provider, a large number of decentralized, i.e. peer-to-peer (P2P) streaming approaches have been proposed [22]. While some of them are meant to operate completely decentralized, also hybrid CDN-P2P approaches, such as Akamai's NetSession [23] have been recently proposed and successfully applied. Using otherwise idle client resources, i.e. upload bandwidth, P2P streaming systems are able to greatly reduce the load on the content providers as well as on CDNs. Especially for small content providers and live streams with hard to predict dynamics in the number of users, P2P streaming remains a promising approach. The key factors that are usually applied to investigate the streaming quality of such decentralized live streaming approaches include the achievable playback delay, i.e. the time between broadcasting at the streaming source and playback at the clients, as well as the caused overhead. Depending on the delivery coordination and built topology, playback delays can vary between less than a second and minutes [22].

In September 2012, Bram Cohen presented a novel P2P streaming system based on the so called screamer protocol [5], specifically targeting live streaming with low delays and low overhead. The approach was filed as a US patent [6] and published as a first implementation, called *BitTorrent Live*[4] (BTLive), in March 2013. According to [6], the specific benefits of the BTLive system are its low latency and low overhead. For content providers considering to use BTLive to broadcast potentially large-scale live events, it is essential to get an in-depth understanding of its properties and limitations. To the best of the authors' knowledge, there currently exists no publicly available study on BTLive's performance characteristics. For P2P streaming, in particular, the tradeoff between performance, in terms of delay imposed by the system, and the costs in terms of server load on the content provider itself, as well as the overhead imposed on the individual clients need to be well understood. As clients are contributing with their resources, they become a crucial part of the system.

To this end, this paper presents a detailed measurement study of the BTLive streaming system conducted since March

[2]https://www.youtube.com/ [Accessed July 30, 2014]
[3]https://www.netflix.com/ [Accessed July 30, 2014]
[4]http://live.bittorrent.com/ [Accessed July 30, 2014]

978-1-4799-6201-3/14/$31.00

2013. The system is currently down as the BTLive developers are preparing for a mobile version of the system. As basis for measurements, the official beta version of BTLive was used, which was accessible on the BitTorrent webpage until February 2014. The goal of this paper is to quantify BTLive's key system characteristics and, thus, answer the following three key questions: (1) How P2P is BTLive? (2) How delay optimized is BTLive? (3) What is the overhead of BTLive?

The remainder of the paper is structured as follows: Section II provides an overview on BTLive and its theoretical background. Section III describes the measurement methodology, while Section IV presents the results of the study. Finally, Section V discusses related work, while Section VI concludes the paper and summarizes the results.

II. BitTorrent Live

A huge variety of P2P live streaming protocols have been proposed over the years (see [22] for an overview). In comparison to those systems, BTLive can be classified as hybrid streaming overlay that applies a number of different mechanisms at the same time. Figure 1 shows the structure of the stream delivery for an example configuration. This approach, which was introduced in [5], could be confirmed during the measurement study presented in this paper. The process conceptually consists of three stages: (1) a push injection of video blocks from the streaming source into the so called clubs, (2) an in-club controlled flooding, and (3) a push delivery to leaf peers outside the individual clubs.

The source divides the video stream into substreams, which is a well-known concept in multi-tree-based streaming [2], [15]. A tracker is used for peer discovery, similar to the concept known from BitTorrent [4]. The peers are divided into clubs, whereby each substream belongs to one of the clubs. The assignment to clubs is done by the tracker when a peer joins the system. A peer belongs to a fixed number of clubs in that it is an active contributor. In all other clubs, the peer is a leaf node and solely acts as downloader. The source plays a special role and always belongs to all clubs to inject the video stream to the individual clubs. Peers strive to establish multiple in-club download connections with members belonging to the same club as well as a single out-club download connection within each of the other clubs. To contribute to the distribution of blocks, peers can establish multiple upload connections to peers within the same club as well as to leaf nodes outside the club. The objective of a club is to spread video blocks of its respective substream as fast as possible to many nodes that can then help in the further distribution process. According to [5], peers can be part of multiple clubs at the same time. This has the advantage of being able to balance upload resources across clubs, but is not required for the approach to work.

As mentioned, peers strive to establish a number of upload and download connections inside their own clubs to help in fastly spreading blocks of the respective substreams. In contrast to pure multi-tree approaches, BTLive does not build up a single, stable datapath within the clubs. Instead, it builds a structure that can be classified as mesh topology, where peers may receive blocks within the club from any of the peers they have a download connection with. While typical mesh-based streaming systems use a pull-based mechanism for a controlled block delivery process, BTLive relies on a push-based mechanism, where peers push newly received blocks to all its upload connections within a club. This way, the costly exchange of blockmaps as well as the requesting of blocks, inherent to pull-based streaming, are avoided. While this implies a significant reduction of delays for the spreading of blocks, it also has a major drawback: it introduces the problem of duplicate block transfers. The reason is that the forwarding process is similar to a flooding of the mesh inside the club. While peers can locally avoid duplicate forwarding of the same block to the same neighbors, they cannot eliminate the high chance of delivering blocks that other peers concurrently send to the same neighbors. Due to the large size and number of video blocks, this can cause a significant overhead, reducing the overall efficiency of the streaming process.

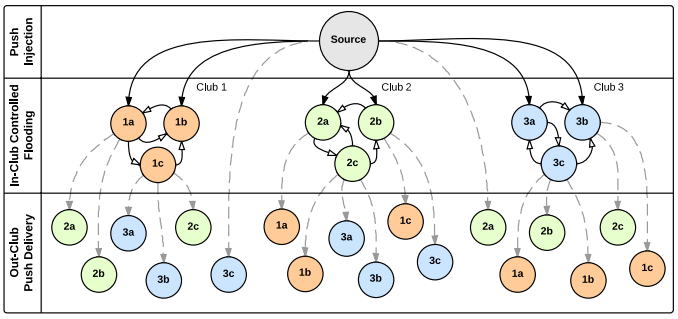


Fig. 1. Three-stage delivery process of BTLive: (1) push injection of substreams into clubs; (2) in-club controlled flooding; (3) push delivery to peers outside the individual clubs.

To mitigate this problem, BTLive includes an additional concept: Every time a peer receives a block within a club, it immediately sends out a message to all other download connections inside the same club to announce the arrival of the block to them. As this message is much smaller than a message including video payload, the chance of sending duplicates can be reduced as neighbors can quickly learn which blocks were already received by the peer and, thus, should not be pushed to the peer a second time. This reduces the chance of duplicates but cannot avoid them. Especially the fast spreading of new blocks and the inherent time dependency between blocks implies a high chance of peers within a club concurrently sending the same block at the same time.

In summary, BTLive's design strives to reduce the streaming delay at the cost of duplicate block transfers. In Section IV, this tradeoff is further discussed and the overhead caused by the beta version of BTLive is investigated under realistic conditions. The latter is an important information to understand the performance and costs of the approach and to compare it to other state-of-the-art streaming solutions.

Theoretical Model

A simplified theoretical model has been derived to describe the processes of data distribution in BTLive and the bandwidth required at the streaming source, i.e. the broadcasting server. The model is derived from the protocol definition and, in particular, the delivery mechanisms that are tightly coupled to the concept of clubs. The purpose of this model is to allow for a worst- and best-case estimation of the required

upload bandwidth at the source. Thus, it can help to describe when the P2P effect can take over, meaning that new peers can completely be served by other peers, not affecting the bandwidth of the source.

For the model, in the following, it is assumed that each peer is a member of exactly one club, while the source is a member of all clubs. For simplification purposes, it is further assumed that all peers have sufficient upload bandwidth.

Two cases are distinguished: In the *static case*, connections remain active once established, while in the *dynamic case* connections with the source can be detached in the process to optimize the overlay structure. The former describes the worst case and the latter the best case in terms of required upload capacity at the source. Depending on the specific implementation of the protocol, overlay structures are expected to result in required capacities in-between these two cases.

1) Static case: The overlay structure resulting from the pure static case is depicted in Figure 2 on the left. In state 1, the first peer (A) joins the swarm[5]. It belongs to club 1 and requires 6 download connections with the source (S), one for each club. When the next peer (B) joins, the first peer provides it with the stream for club 1. The remaining streams are served by the source. The third peer (C) receives the streams for club 1 and 2 from the first and second peer and the remaining streams from the source. This case is called *static* as existing connections are not replaced by connections to other peers that join later. In this case, a peer is only served in full P2P fashion if at least one peer in each club already exists.

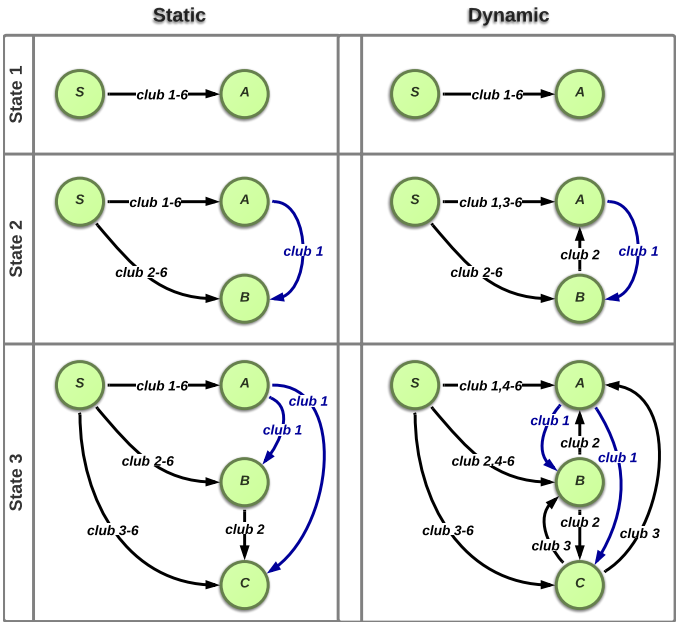


Fig. 2. Comparison of the two cases for the first three joining peers.

The minimum bandwidth required at the source for the static case can be calculated using the following non-closed or closed-form expressions:

$$bw_{\text{static}} = b_{\text{trans}} \times \sum_{i=0}^{c-1}(1 - \frac{i}{c}) = b_{\text{trans}} \times \frac{c+1}{2} \qquad (1)$$

[5]The term "swarm" refers to all active peers participating in the overlay.

where c is the total number of clubs and b_{trans} the bitrate required for transmitting the video stream using the BTLive message format. The latter can be calculated by adding the BTLive protocol overhead to the average video bitrate.

For 6 clubs, bw_{static} would result in 3.5 times the transmission bitrate b_{trans}. This factor increases to 6.5 for 12 clubs. The ratio of the number of clubs to the required resources in the static case is illustrated in Figure 3. As expected, it shows that the required source upload bitrate increases steadily until there is one member for each club. Subsequently, it levels off and the peers are fully providing the streams for all additional peers that join thereafter.

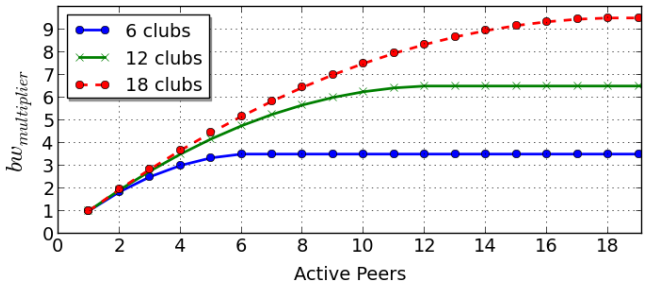


Fig. 3. Required source upload bandwidth in the static case. The absolute value is calculated by multiplying this factor by the transmission bandwidth.

2) Dynamic case: This case is depicted in Figure 2 on the right. A suitable replacement of connections results in less bandwidth required at the source. Thereby, it is the most ideal process for minimizing required source bandwidth. In state 1, (A) joins and subsequently gets all clubs from (S). In this state, there is no difference between the two cases. The difference comes into effect in state 2, where (A) terminates its connection for club 2 and establishes the same connection with (B) instead, which is member of club 2 and, thus, can help to offload the source.

In the *dynamic case*, the minimum bandwidth required at the source can be calculated as follows:

$$bw_{\text{dynamic}} = b_{\text{trans}} \times \sum_{i=0}^{\lceil \frac{c}{2} \rceil - 1}(1 - \frac{2i}{c}) \qquad (2)$$

For brevity reasons, the closed-form variant is omitted here.

For 6 clubs, bw_{dynamic} results in 2 times the transmission bitrate b_{trans}. The factor increases to 3.5 for 12 clubs. The ratio of the number of clubs to the required resources in the dynamic case is illustrated in Figure 4. It shows that the required source upload bitrate increases steadily until there is one member for $\lceil \frac{c}{2} \rceil$ clubs. Subsequently, the peers start to release the source and the required source upload bitrate decreases until there is one member for each club. It then remains at a level of one time the transmission bitrate and the P2P effect completely takes over all remaining upload.

The theoretical model shows that the minimum number of peers necessary for the P2P effect to start as well as the minimum upload bandwidth required at the source both strongly depend on the number of clubs. The higher the total number of clubs, the more upload bandwidth is to be provided by the source and the more peers are required for

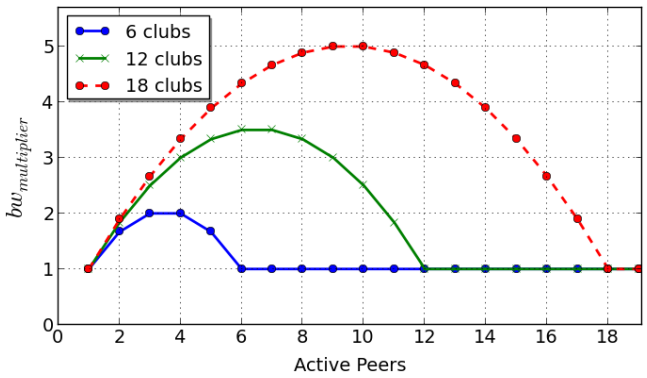


Fig. 4. Required source upload bandwidth in the dynamic case. The absolute value is calculated by multiplying this factor by the transmission bandwidth.

the P2P effect to take over. In the remainder of the paper, this inherent limitation of BTLive is referred to as *source bottleneck problem*. Following the theoretical model, this problem leads to increased server bandwidth requirements in scenarios where the number of peers is smaller than the number of clubs. As the number of clubs is a fixed configuration parameter of the BTLive system, it cannot easily be changed during runtime. Furthermore, having a minimum number of clubs is essential to maintain a multi-tree topology with all its benefits [15]. Thus, an adequate choice for this parameter is important and can be highly dependent on the target scenario.

III. Measurement Methodology

The Emanicslab testbed[6] with 20 nodes distributed all over Europe was used to run the streaming source and the clients during the study. Depending on the number of peers involved in the individual measurements, all available sites of the testbed were used. For experiments exceeding the number of physical nodes, multiple peers were run at individual machines within the testbed. The tracker software was not published and, thus, could only be run by BitTorrent Inc. Based on the observed IP address during the study, a single tracker seemed to be run most likely using Amazon EC2 web services[7].

The network traffic at the individual clients as well as the source was captured and studied to derive statistics related to individual peer connections, different packet types, and the protocol flow. Besides, timing-related measurements were conducted to study the streaming delay properties of BTLive. Hereby, the streaming delay is defined as the difference in time between the source sending out a media block and an individual client receiving it. This can be a block that was forwarded directly in only one hop from the source to the client or a block that was propagated throughout the streaming overlay over several hops. Therefore, it describes how long a client has to wait until content is available at its side and can be handed over to the media player for playback. The actual playback delay can be longer, depending on the decoding process and the playback state of the video player. As the used player software could not further be investigated and the playback as well as the buffering strategies are usually out of control of the streaming overlay, it is not considered here.

A. Network traffic capturing and analysis

Figure 5 schematically shows the measurement setup across different entities used as vantage points in the study. The streaming software was provided by BitTorrent Inc. as binary that is to be installed by the user and includes the streaming client only. For video playback, the Adobe Flash Player is used, communicating with the local streaming client using RTMP. A client usually visits a channel-specific webpage, where the Flash-based video player is embedded, initializing the connection to the local streaming client and passing it the required information to connect to the channel's swarm. The information includes a channel identifier as well as the contact information of the tracker, which are embedded in the webpage. To be able to run the client software on the headless nodes of the testbed, the tool RTMPdump[8] was used to emulate a video player that is connected to the streaming client and to dump the received video stream to a file. This was done to be able to check the playback quality of the individual clients.

TCPdump[9] was used as measurement tool to capture traces of the incoming and outgoing BTLive network traffic as well as the local RTMP traffic for later analysis. The BTLive traffic is further analyzed using a Wireshark plugin[10], which was developed for this purpose[11]. The RTMP traffic is used to determine the streaming delay. Moreover, the payload of the BTLive traffic needed to be analyzed in detail, as the BTLive protocol structure was not published at the time of this study. Nevertheless, the most important parts of the packet format were identified by using the available information from [5] and an indepth study of recorded network traffic traces. This way, the role of the different message types and relevant fields of these messages could be decoded.

B. Measuring streaming delay

For the streaming delay measurement, first, the clock differences of the nodes were determined using NTP at the beginning of each evaluation run. Using the RTMP traffic traces of the streaming source as well as the individual clients, different methods were investigated to determine the streaming delay. It turned out that the payload of the captured audio packets allows for a reliable identification of individual data blocks within the media stream. As the audio is played back synchronously with the video, this showed to be a good way to study the delay of the overall streaming process in a very fine-granular manner. Therefore, the RTMP traffic was processed using a custom-built software tool, comparing and matching the individual audio packets sent out by the source and the ones received at the individual clients. Using the time difference information between the machines and the timestamps captured within the TCPdump traces, the delay could be calculated for each individual packet. To make sure that matching of audio packets works correctly and does not falsely match duplicates in the stream, duplicates were recorded and excluded during

[6] http://www.emanicslab.org/
[7] https://aws.amazon.com/de/ec2/
[8] http://rtmpdump.mplayerhq.hu/
[9] http://www.tcpdump.org/
[10] http://www.wireshark.org/
[11] http://www.ps.tu-darmstadt.de/research/btlive

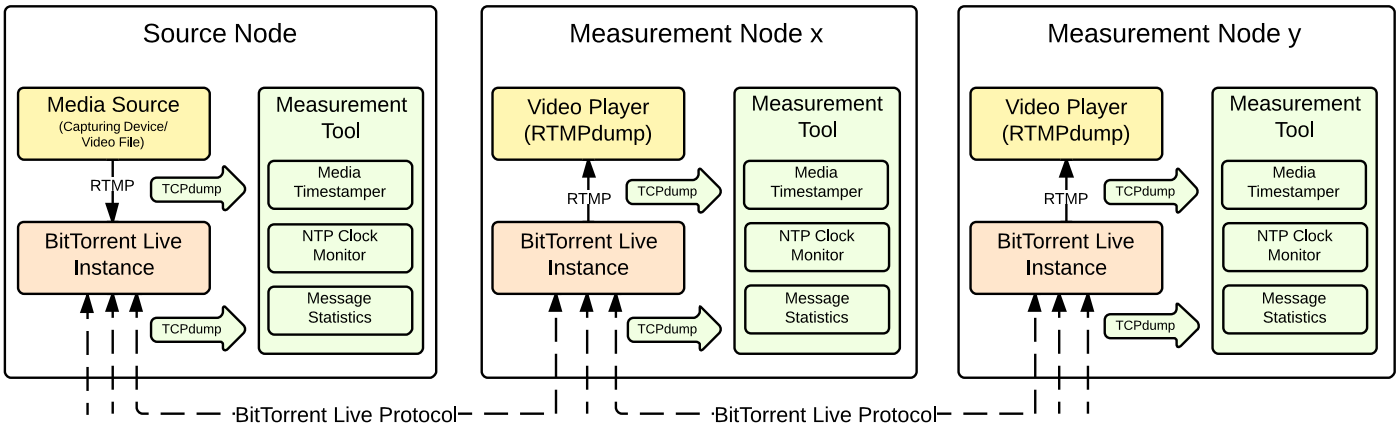


Fig. 5. The setup of measurement tools across the different vantage points of the measurement: the streaming source and the measurement nodes.

the processing of the data. Duplicates can occur, e.g., if a short video is broadcasted in a loop or a video includes exactly the same audio sample multiple times. After identifying this problem and using a video longer than the duration of the evaluation runs, duplicates were avoided completely. As presented in the evaluation, using this method, in average between 10,000 and 12,000 matching samples per client in a 5-minute streaming session were recorded, allowing a detailed and accurate study of the delay characteristics.

IV. Measurement Results

All experiments were conducted using an own channel, where a test video was broadcasted by an streaming source running on a dedicated host. The test video[12] was encoded with an average bitrate of 471 kbit/s. Depending on the experiment, different numbers of peers were run on Emanicslab machines. For the measurements, the latest published version (0.4.12.335) of the BTLive client was used. Using the Linux *TC* (traffic control) tool, the upload bandwidth of the source was limited to 4 times the average video bitrate, as recommended by BitTorrent Inc. as minimal upload bandwidth for the source. Experiments were repeated 10 times and 95% confidence intervals are reported for all averaged values.

A. How P2P is BTLive?

Even if P2P live streaming cannot work without a streaming source, the goal of any P2P approach is to shift load away from the source to the peers, i.e. the amount of data delivered by the source should be small in comparison to the peers. To understand how well BTLive achieves this goal, different system configurations with respect to the swarm size were studied, comparing their traffic characteristics. In all cases, the traffic measurements were conducted at a single peer that enters the system after all other peers already joined the swarm. This way, the start-up phase of bootstrapping the overlay is skipped as the goal was to study the streaming process after the system stabilized. The overlay traffic was recorded at the measurement peer for five minutes, after which the peer left the system. The five different swarm sizes studied are labeled according the number of peers present in the swarm, excluding the source and the measurement peer itself: 0, 2, 10, 50, and 90.

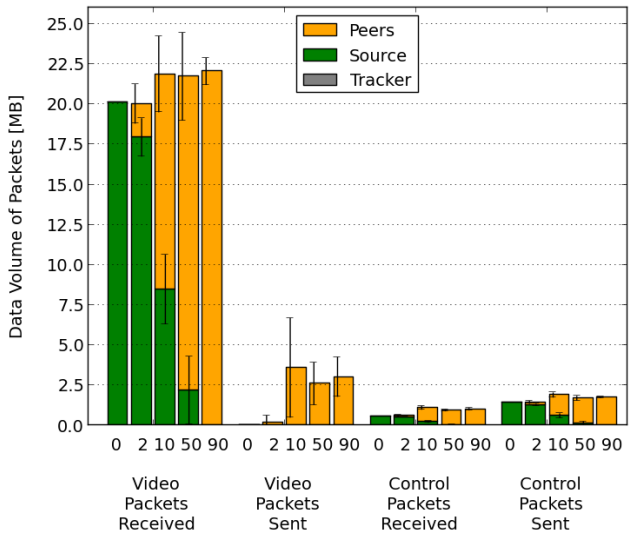
[12]http://www.bigbuckbunny.org/

As mentioned before, for scenarios with more than 20 peers, multiple peers were run on the individual testbed machines.

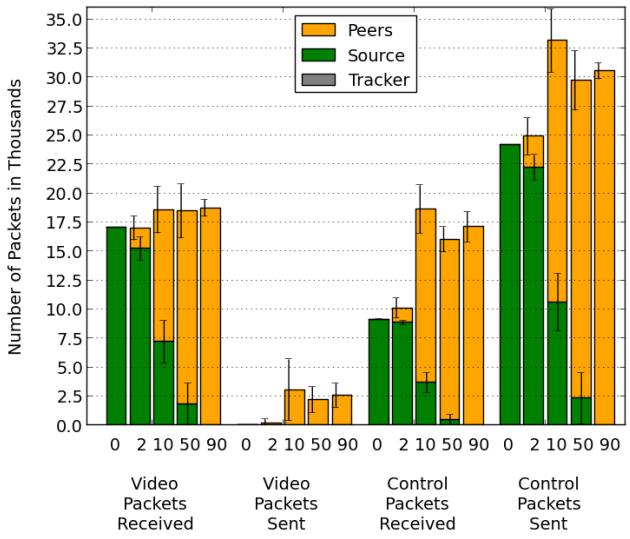
First of all, the number of clubs used by the overlay was studied based on a sample BTLive traffic trace. The trace revealed that in the observed BTLive version the number of clubs is configured to be 6, a number much smaller than expected, since from [5] it seemed that a number of 12 clubs would be a desired configuration. Besides, the traces showed that, as expected, the source is an active contributor to all of the 6 clubs as it initially injects the data into the clubs. Furthermore, it turned out that the peers exclusively belong to only one club, which reveals that the advertised load balancing mechanism [5] to efficiently distribute heterogeneous peer resources among clubs is not implemented so far.

In the context of the previous finding, the observed reduction in the number of clubs could be explained by the source bottleneck problem that was identified in Section II. A study of the channel popularity of the beta version of BTLive over several months, which was conducted in parallel to this measurement study, showed that often most channels were not used at all or only by a small number of clients. While for large-scale streaming scenarios a higher number of clubs is desirable [6], for scenarios where the number of peers is most of the time smaller than the number of clubs, the source bottleneck problem would clearly hinder the P2P effect and force the source to handle most load of the system. This could explain why the developers configured the protocol to use only 6 clubs. Furthermore, this implies that the number of clubs has to be chosen carefully, depending on the scenario.

Building up on these findings, Figure 6 shows the relation between video data served by the source and peers, for the five different swarm sizes as described above. Figure 6a and 6b depict the data volume and total number of BTLive packets, respectively, sent and received by the measurement peer and averaged over 10 repetitions of the measurement. The peer received on average roughly the same amount of data and packets for all five swarm sizes. As expected, for a swarm size of 0 and 2, the measurement peer has no or only limited chances to contribute to the video packet distribution. Almost all packets are coming from the source. With no other peers in the channel, the measurement peer on average received 20.1 MB of video packets (17.1 thousand) and did not sent any video packets. It further received 0.5 MB (9.1

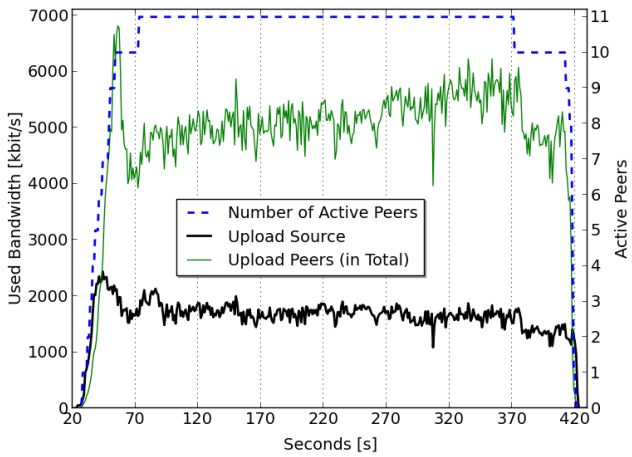


(a) Data volume of BTLive packets (0.95 confidence interval).

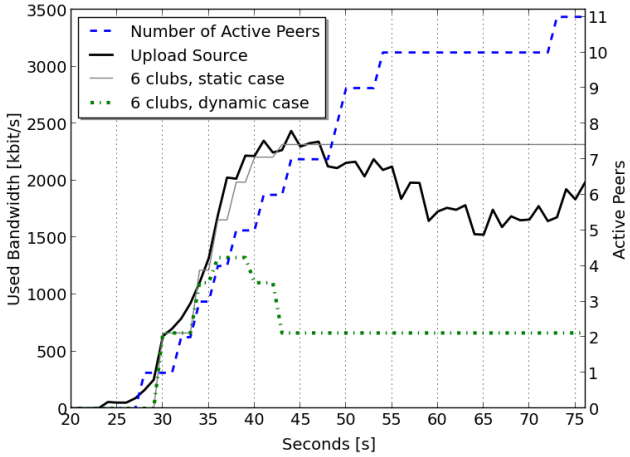


(b) Number of BTLive packets (0.95 confidence interval).

Fig. 6. Traffic characteristics for the 5-minute streaming session as observed at the measurement peer for different swarm sizes.



(a) Comparison of upload bandwidth contributed by source and peers.



(b) Upload bandwidth contributed by the source within first 75 seconds.

Fig. 7. Upload bandwidth characteristics over time for the complete streaming session, averaged over 10 measurements.

thousand) control packets from the source and sent 1.4 MB (24.2 thousand) control packets to the source. The average size of the video that was received and saved by RTMPdump was 16.4 MB. The difference between the video data and the volume of received video packets of around 3.7 MB (18.4%) is assumed to be caused by message overhead (i.e. roughly 5% for video packet headers) and the transport format used for video data transmission. The latter is supported by the observation that video data was transformed by RTMPdump after receiving and before storing it for later investigation.

In all settings, the data volume for sending and receiving control packets was below 3 MB, which roughly refers to less than 11% of the overall sent and received data. For the amount of received video data, the expected P2P effect is clearly visible with an increasing swarm size. If the peer is the only peer in the swarm, it naturally receives only packets from the source. With 2 other peers in the swarm, on average about 11% of the video data is delivered by peers. With 10 other peers, the share already increases to about 61% in average served by peers. With 50 peers, on average about 90% are served by peers, while with 90 peers, the measurement peer did not receive any video data from the source. The communication with the tracker is negligible and thus not visible in the figures. For the number of received and sent video packets, the trends look similar. A major difference is visible for the control packets. While they only make up for less than 11% of the data traffic, in all cases more control packets are sent than video packets are received. Besides, looking at the confidence intervals, the number of received control packets reaches a level that is not significantly different to the number of received video packets. This rather high number of control packets is assumed to be a result of the screamer protocol [6] where the arrival of video packets is announced to all in-club download connections to avoid duplicate video packet transmissions.

Figure 7 provides an alternative view on the streaming process as it shows the contribution of upload bandwidth by the source and the peers over time for a swarm size of 10 peers (plus the measurement peer), averaged over the 10 repetitions. Here, the traffic was captured and aggregated at all peers to provide a swarm-wide view on the used resources. Figure 7b shows the upload bandwidth used at the source for only the first 75 seconds of the measurement. Both figures show that the average upload rate of the source increases by each newly joining peer until it reaches its maximum of 2,436 kbit/s after 44 seconds, i.e. after the 7th peer joined the channel. Subsequently, the upload rate of the source decreases, even though 3 more peers join the swarm. At the same time, the upload rate of the active peers is increasing, especially directly after peers number 8 and 9 join. It reaches its maximum at 6,816 kbit/s after 56 seconds, with 10 peers in the channel. The peak in the contributed upload bandwidth at the source confirms the theoretical model in Section II, according to which for 6 clubs the minimum upload bandwidth is between 2 (dynamic case) and 3.5 (static case) times the transmission bitrate, which on average was about 662 kbit/s in this case.

Both the dynamic and the static case are also depicted in Figure 7b. The comparison of the actually observed source bandwidth with the two theoretically derived curves shows that the BTLive beta version closely follows the worst (static) case until second 46. Afterwards, it slightly decreases, which is also visible in Figure 7a for the rest of the measurement duration. This shows that connections by the source seem to be only occasionally replaced by connections from other peers. Thus, the P2P effect could be easily improved if the source would more aggressively close connections to clubs that are served by multiple connections, forcing peers to take over more load. If this should actually be done, is up to the content provider as surplus source bandwidth could also help to cope with peer churn and temporal inefficiencies in the streaming process.

Furthermore, the upload bitrates in an exemplary period of the steady state (i.e. between seconds 310 and 369) were determined. This period of time is considered steady as it lies between the initial decline on the source and the first leave of a peer. Any other period between about seconds 70 and 370 could have been used here. In this state, with 11 peers in the swarm, the total upload bitrate was 7,285 kbit/s, which on average corresponds to a 662 kbit download bitrate for each of the 11 peers in the channel. The average upload bitrate of the source in steady state was 1,648 kbit/s (22.62% of the load), while the average total upload bitrate of all peers in steady state was 5,637 kbit/s (77.38%).

In a recent interview [9], an employee of BitTorrent Inc. stated that the P2P effect takes over at around 10 to 20 concurrent viewers and that from there on it scales with a growing number of users with no extra cost at the source. Indeed, the measurement results showed that the peers take over more load after a minimum number of 6 peers are present to help spreading the blocks of the stream to all 6 clubs. Before that, the identified source bottleneck problem hinders the P2P effect to take over. If peers do not have enough upload bandwidth to forward the blocks of their club, clearly the effect takes more peers to take over. Yet, it was also shown that the P2P effect could even be improved by more aggressively reducing contributions by the source.

B. How delay optimized is BTLive?

BitTorrent Inc. recommended a source upload bandwidth of at least four times the average video bitrate. Thus, for the delay study, an experiment was conducted where the source upload bandwidth was limited to 2,000 kbit/s (roughly 4×471 kbit/s). Especially for configurations with low peer bandwidths, this setting showed to be too low to maintain a stable streaming process. This can be explained using the model presented in Section II. Following the worst (*static*) case of the model and the observed parameters of BTLive, the source should be configured to provide at least 3.5 times the transmission bandwidth (662 kbit/s), resulting in 2,317 kbit/s. Therefore, a second experiment was conducted with an increased source bandwidth of 2,400 kbit/s, which showed a stable streaming behavior across the measurement peers. This implies that the recommendation for the minimum bandwidth should be higher.

To investigate BTLive's delay characteristics, a number of experiments with changing peer bandwidth configurations were conducted. For realistic network delays, 10 peers were deployed on dedicated machines at distinct Emanicslab sites. The observed average streaming delays for the different configurations are depicted in Figure 8. Additionally, the figure includes the number of matched samples that were used to calculate the average delays. The latter was added to verify that a sufficient number of samples was matched to draw valid conclusions. Although the number is slightly lower for the measurement with a low source and peer bandwidth, the total number of 10,000 samples is still on a high level.

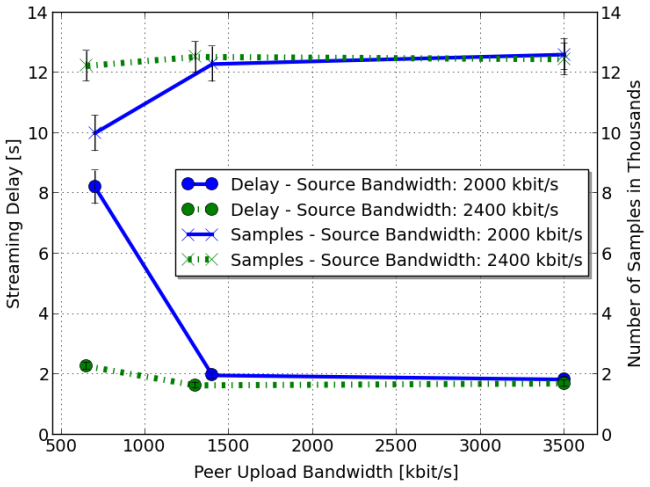


Fig. 8. Average measured streaming delay and number of matched samples.

For the first measurement series with an reduced source upload bandwidth (2,000 kbit/s), an interesting effect can be observed when also limiting the peers to a low upload bandwidth (700 kbit/s). The average delay (around 8 seconds) was significantly higher than for all other configurations. At the same time, a drop in the number of matched samples by around 2,000 samples was observed. An in-depth analysis of the captured data showed that the collected traces were significantly smaller than the ones of the other measurements, reducing also the number matching samples. Both can be explained by peers either not receiving the complete stream due to the scarce resources, or by peers leaving the swarm

due to bad performance. By increasing the upload bandwidth at the source to 2,400 kbit/s, the average streaming delay significantly dropped to around 2 seconds, although the peer's upload bandwidth was slightly decreased to 650 kbit/s in the second measurement series. For all other configurations, the average streaming delay was observed to be below 2 seconds.

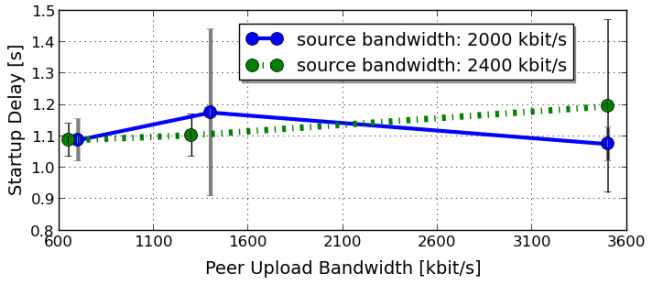


Fig. 9. Average startup delay over the different configurations.

Besides the average streaming delay, another important aspect is the startup delay as it greatly influences the abandonment rate of users in video streaming systems [18]. Startup delay was defined as the time between a peer sending a first packet (i.e. the join message to tracker) and the arrival of the first media packet. This definition only captures the overlay characteristics. The real startup delay observed by the users is higher and depends on the video player, its buffer management, and the implemented playback policy. Figure 9 shows that the observed average startup delay is very low and stable, ranging between roughly 0.6 and 1.2 seconds for 95% of the measurements across all configurations. However, in all configurations single outliers were observed were the startup delay was significantly higher, ranging up to 15.5 seconds. To study this in more detail, Figure 10 shows the distribution of the measured delays over all samples. Here, the difference between the configuration with scarce source upload capacities becomes clearly visible. While for all other cases, the delays spread between 0.7 and 4.6 seconds, for this configuration, it varies much more and spreads from roughly 2 to 17 seconds. This clearly shows that the streaming process was suffering due to insufficient resources at the source, implying that the minimum capacity specified by BitTorrent Inc. is too low for cases were peers have low capacities as well. Some more measurements were conducted in which the source bandwidth was set to even lower values than 2,000 kbit/s, resulting in highly unstable streaming processes. This observation initially led to the theoretical considerations presented in Section II, providing a good explanation of the system behavior.

According to BitTorrent Inc. [9], the streaming delay averages around five seconds, regardless of the swarm size. With 10 peers in the channel, the lowest average latency observed was 1.631 seconds in the case of 2,400 kbit/s source and 1,300 kbit/s peer upload bandwidth. In sum, the low delay properties of BTLive could be confirmed in the measurement study for small swarm sizes, as long as the source capacity was higher than the theoretically derived threshold.

C. What is the overhead of BTLive?

Overhead in video streaming is mainly a result of control traffic or duplicate transmissions due to lack of coordination. For P2P overlay maintenance and data exchange coordination, control traffic is inevitable and can greatly differ depending on the applied topology and delivery concepts. In comparison to the delivered media data, the volume of control traffic is usually negligible. Nevertheless, the large number of control messages can cause a significant load on the network.

Besides control traffic, duplicate packets can significantly contribute to overhead. As described earlier, due to its aim to reduce streaming delay, BTLive follows a controlled flooding approach in which peers send out announcements of packet arrivals to other in-club peers as soon as a packet was received. This approach accepts that media packets may actually be delivered multiple times to the same peer. To show that this is actually the case, Figure 11 depicts the data volume caused by both control traffic and duplicate media packets. For the same configurations as here, Figure 6 above showed how the data volume compares to the number of packets. It is notable that the increase in volume and number of video packets due to duplicate video packets is very well visible for 10, 50, and 90 peers. Distinguishing further between unique and duplicate video packets, unique video packet of about 20 MB on average per peer were observed with only small variation across all measurements. In contrast, the data volume caused by duplicate media packets shows more variation as the steep increase between a swarm size of 2 and 10 peers shows. It is apparent in Figure 11 that for a swarm size of 10 peers or higher, the overhead from duplicates ranges between 1.7 and 1.9 MB, relating to roughly 8% of the video data. At the same time, the overhead from control traffic ranges between 0.5 and 1.1 MB. In sum, the overhead, including both duplicates and control traffic, on average accounts for roughly 12% of the overall traffic, where around 65% of the overhead is caused by duplicates. Compared to other state-of-the-art systems this overhead is very high. In [21] it was shown that typical mesh and tree-based approaches exhibit roughly between 0.5 and 5.5% overhead and that a hybrid approach can achieve less than 1% overhead, even for challenging scenarios. This clearly shows the price BTLive pays for low streaming delays.

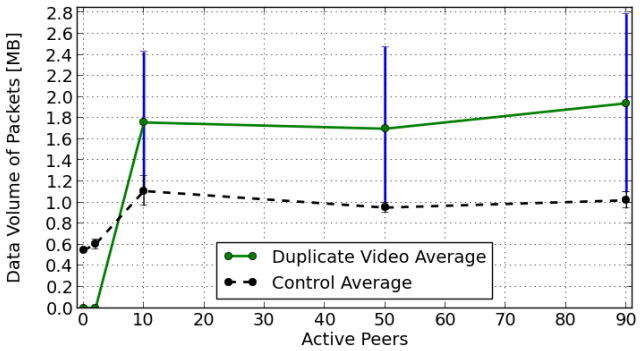
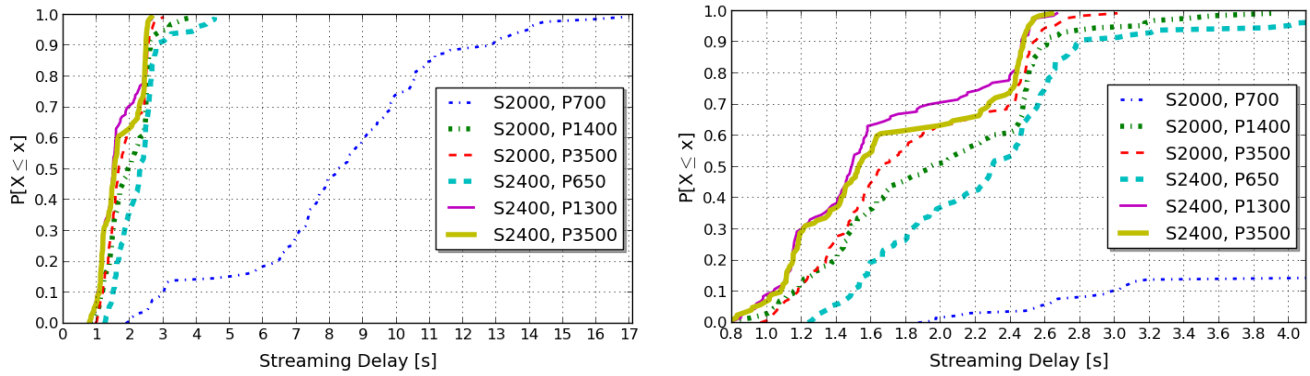


Fig. 11. Average data volume of control traffic and duplicate video packets observed at the measurement peer in scenarios with different numbers of peers.

To further study the overhead, the traffic caused inside and outside clubs was distinguished. Figure 12 shows the different classes of overhead and, for comparison, the unique video packets delivered. The result supports the earlier findings for the cause of overheads. For each club that a peer is not a member of, the peer has one out-club download connection.



(a) Distribution of streaming delay

(b) Distribution of streaming delay (zoomed in)

Fig. 10. Distribution of streaming delays for the 6 different configurations. S2000 and S2400 depict source bandwidth configurations with 2,000 and 2,400 kbit/s. P700, P1400, P3500, etc. depict different peer bandwidth configurations, respectively.

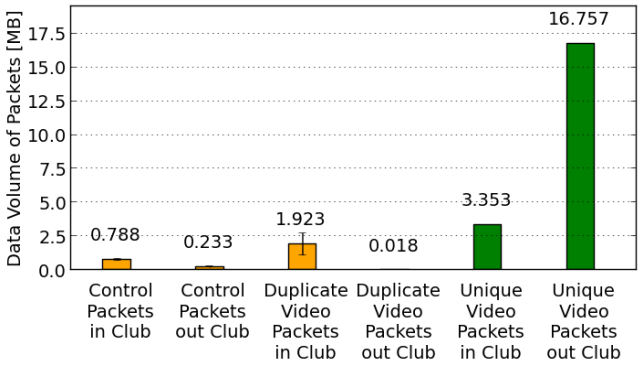


Fig. 12. Average data volume for different traffic classes, inside and outside clubs captured at the measurement peer in the scenario with 90 peers.

Using these connections, most of the unique media packets are delivered, which on average account for 16.8 MB, while almost no duplicates were observed. For the club to which the peer is an active contributor, it was observed that it has between 2 and 3 in-club download connections. Here, it is possible that multiple in-club uploaders send the same media packet at the same time, causing a duplicate transmission for the receiving peer. Therefore, both unique and duplicate video transmissions were observed for the in-club case. On average, about 3.4 MB of unique and 1.9 MB of duplicate media packets occurred. The latter translates to roughly 8.6% of all received media packets. CDFs are not shown due to space restrictions. They showed very stable results for all traffic classes except for the duplicate in-club packets, which varied between 0.06 and 3.29 MB. This means that in some cases, the data volume caused by in-club duplicates were as high as for the unique media packets, even though around 0.79 MB of control packets were received inside the club, mainly caused by packets that announce the arrival of new media blocks. This is observable in Figure 6b, which shows that the controlled flooding approach inside the mesh-based clubs imposes significant overhead, compared to the media data transmitted within the club. Out-club deliveries happen without any duplicates, as expected for a pure push-delivery over a tree topology.

In sum, while BTLive avoids exchanging block maps in the mesh-based clubs, it compensates for that by sending announcement packets for the arrival of blocks, resulting in a large number of control packets being sent. In spite of this delivery coordination, BTLive causes significant overhead in form of duplicate video packets that are avoided by other approaches not combining mesh- and push-based delivery.

V. Related Work

To the best of the authors' knowledge, there currently exists no publicly available measurement study for BTLive. However, various studies have been conducted over the last decade analyzing other P2P streaming systems. Most of these studies analyzed PPLive or PPStream.

Vu et al. [20] and Hei et al. [13] conducted active measurements to study overlay and streaming session characteristics of PPLive. The first work studied graph properties of the overlay and, e.g., shows that PPLive overlays up to a certain size can be described as random graphs and that the average peer degree is independent of channel populations. The second work shows that PPLive causes long start-up and playback delays, ranging from several seconds to a couple of minutes.

Liang et al. [14], in contrast, conducted passive measurements of PPStream during the 29th Beijing Olympics to study streaming characteristics. The authors show that the Olympics' channels differed from the rest of the channels in terms of playback delay and smaller scheduling units to achieve a timelier and more reliable, yet less efficient delivery of the streams. Another passive measurement study was conducted by Gao et al. [12], showing that both systems PPLive and PPStream can provide an excellent viewing experience for popular channels but are rather inefficient for unpopular channels. Alessandria et al. [1] conducted a passive measurement study for different commercial streaming systems, namely PPLive, SOPCast, TVants, and TVUPlayer. The authors observed that all applications avoid bad network paths and carefully select neighbors. Furthermore, they found that the behavior of the peers can get aggressive if there is a bottleneck which affects all peers, for example at the access link. A more recent study from 2013 on SopCast by Vieira et al. [19] observed that SopCast channels tend to have users in the order of hundreds but can become much larger during special events. Their

measurements show that almost 75% of all packets exchanged between peers were control messages.

Some of the studies discussed above used a similar approach as the one applied here for measuring a given P2P streaming system under realistic conditions. Moreover, this paper studied BTLive in a controlled environment, i.e. the source and all peers in the study were under the authors' control. This allowed for an in-depth understanding of the streaming process primitives of this novel streaming protocol.

VI. Discussion and Conclusion

As stated in the beginning, the goal of this paper was to answer three key questions related to the characteristics of BTLive. The first question was: how P2P is BTLive? This paper shows that the P2P effect takes over with an increasing number of peers in the swarm, above a minimum number of 6 peers (equal to the number of observed clubs). In addition, the used bandwidth at both source and peers was studied over time. The required upload resources at the source were described by a theoretical model, matching the observed source bottleneck with a small number of peers. The model includes both an estimate for the lower and the upper limit for the required upload bandwidth at the source over time. The results show that there exists great potential to reduce the load at the source by more aggressively dropping connections. Finding the right balance between sufficient capacity and adequate overprovisioning is the key in this aspect. The system should be able to cope with peer churn and temporal inefficiencies in the delivery process, while keeping the costs for the content provider as low as possible.

The second question asked how delay optimized BTLive is. To answer it, a set of controlled experiments with bandwidth limitations at the source and peers to limit the delivery paths were conducted. It is shown that BTLive is able to maintain both very small startup delays of less than 1.2 seconds for the considered scenarios as well as small streaming delays, as long as the source contributes a minimum bandwidth. In this context it was shown that the recommended minimum source bandwidth of 4 times the video bitrate was not enough to provide a stable system performance, which also can be explained by the proposed theoretical model. For experiments with sufficient source capacities, BTLive was confirmed to have low delays for the studied configurations. Yet, the delay properties require further investigation for larger swarms.

The final question was: What is the overhead of BTLive? To answer this question, the exchanged streaming traffic was studied in more detail. It could be shown that the advertised low delays that can be achieved by the system comes at a high cost, in terms of control and video traffic inside the clubs which is caused by the controlled flooding approach. The announcements used within the clubs to avoid the transfer of duplicate media packets between peers were not able to avoid duplicates. In some cases the number of unique and duplicate video packets delivered to a peer within the club was even the same, i.e. double the data volume was produced inside the club as desired. In future measurement studies it is to be investigated how this problem affects streaming in larger swarms. Especially the envisioned application of BTLive in mobile environments could require a rethinking of the controlled flooding approach inside clubs.

Acknowledgment

This work has been funded in parts by the European Union (FP7/#317846, SmartenIT and FP7/#318398, eCOUSIN) and the DFG as part of the CRC 1053 MAKI.

References

[1] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo, "P2P-TV Systems under Adverse Network Conditions: A Measurement Study," in *IEEE INFOCOM*, 2009.

[2] M. Castro *et al.*, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *ACM SIGOPS*, 2003.

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

[4] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer systems (P2PECON)*, 2003.

[5] ——, "How to Do Live P2P Video Streaming," 2012, Keynote Speech at IEEE International Conference on Peer-to-Peer Computing. [Online]. Available: http://www.p2p12.org/program/keynote-speakers

[6] ——, "Peer-to-Peer Live Streaming," Patent 20 130 066 969, March, 2013. [Online]. Available: http://www.freepatentsonline.com/y2013/0066969.html

[7] Die Medienanstalten, "Digitisation 2013 - Broadcasting and the Internet - Thesis, Antithesis, Synthesis?" Tech. Rep., 2013.

[8] C. Diot, B. N. Levine, B. Lyles *et al.*, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network*, vol. 14, no. 1, pp. 78–88, 2000.

[9] T. Dreier, "BitTorrent Recruits Testers for Live Video Platform," Oct. 2013, Interview with Tim Leehane, BitTorrent Inc., Streaming Media East Conference. [Online]. Available: http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/BitTorrent-Recruits-Testers-for-Live-Video-Platform-92757.aspx

[10] Ericsson ConsumerLab, "TV and Video - An analysis of evolving consumer habits." Tech. Rep., 2012.

[11] ——, "TV and Media - Identifying the Needs of Tomorrow's Video Consumers," Tech. Rep., 2013.

[12] G. Gao, R. Li, W. Xiao, and Z. Xu, "Measurement Study on P2P Streaming Systems," *Springer Journal of Supercomputing*, vol. 66, no. 3, pp. 1656–1686, Jun. 2013.

[13] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.

[14] W. Liang, R. Wu, J. Bi, and Z. Li, "PPStream characterization: Measurement of P2P live streaming during Olympics," in *IEEE Symposium on Computers and Communications*, 2009.

[15] Y. Liu, Y. Guo, and C. Liang, "A Survey on Peer-to-Peer Video Streaming Systems," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, 2008.

[16] E. Nygren, R. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-performance Internet Applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[17] Sandvine, "Fall 2013 Global Internet Phenomena Report," 2013.

[18] R. K. Sitaraman, "Network Performance: Does It Really Matter To Users and by How Much?" in *IEEE COMSNETS*, 2013.

[19] A. B. Vieira, A. P. C. da Silva, F. Henrique, G. Goncalves, and P. de Carvalho Gomes, "SopCast P2P live streaming: live session traces and analysis," in *ACM Multimedia Systems Conference on (MMSys)*, 2013.

[20] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Measurement and Modeling of a large-scale Overlay for Multimedia Streaming," in *ACM QSHINE*, 2007.

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

[22] 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.

[23] M. Zhao, A. Chen, Y. Lin, and A. Haeberlen, "Peer-Assisted Content Distribution in Akamai NetSession," in *ACM IMC*, 2013.