Journal of Interconnection Networks
Vol. 13, Nos. 1 & 2 (2012) 1250003 (19 pages)

DOI: 10.1142/S021926591250003X

World Scientific
www.worldscientific.com

A DYNAMIC JITTER-CONTROLLED TREE-BASED P2P NETWORK TOPOLOGY FOR MULTIPOINT MULTIMEDIA APPLICATIONS

ALFIO LOMBARDO[*], ANTONIO MASTO[†], DIEGO REFORGIATO[‡] and GIOVANNI SCHEMBRA[§]

Dipartimento di Ingegneria Elettrica, Elettronica e Informatica (DIEEI),
University of Catania, V.le A. Doria 6,
Catania, Italy, 95125
[*]*alfio.lombardo@dieei.unict.it*
[†]*antonio.masto@dieei.unict.it*
[‡]*diegoref@dieei.unict.it*
[§]*schembra@dieei.unict.it*

Received 26 September 2012
Revised 29 November 2012

In the last few years peer-to-peer (P2P) systems have gained ground for multipoint video content distribution over IP networks. P2P technologies give new opportunities to define an efficient multimedia streaming application, but at the same time they involve a set of technical challenges and issues due to the best-effort service offered by the underlying Internet, and its dynamic and heterogeneous nature. Stringent requirements in terms of end-to-end delay for real-time applications motivates the choice of a tree-structured topology against other topologies that have been introduced in the last research works, but mainly aimed at non-real-time services like video on demand and live streaming. The target of this paper is to present a platform for multipoint multimedia transmission based on a tree overlay network with jitter control and to show through experiments on real environment that our platform performs better than a traditional tree overlay network system in terms of PSNR, frame loss and playout frozen time.

Keywords: QoS; P2P; multimedia; jitter; video.

1. Introduction

A lot of research work has been reported in the previous literature mainly regarding transmission of video on demand and live streaming video flows in multipoint fashion. P2P video streaming systems have already achieved a number of large-scale deployments, accommodating tens of thousands of simultaneous users.[1] Some recent

works[2–5] gave us useful hints and suggestions for the development of the system we propose in this paper. The SVC encoding[15] is a recent encoding that allows the construction of bit-streams that contain sub-bit-streams including one known as "base layer". In Ref. 6, a multipoint video broadcast framework over a heterogeneous content distribution P2P network has been proposed. In the proposed system the source generates the video flow by using a layered encoder, specifically an MPEG-4/FGS encoder. A FGS (fine granularity scalability) (see Refs. 6 and 7) stream has only two layers: a base layer that must be received to make possible video decoding, and an Enhancement layer, indicated as the FGS layer, which can be delivered optionally where bandwidth is available. FGS allows the source to adjust the relative sizes of both Base and FGS layers, therefore allowing the Base layer to arrive at all the destinations as a whole. On the other hand, the FGS layer can be broken up and the decoder may decode any portion of it. The source or any intermediate node is responsible to control the size of both Base and FGS layers. Although not widely applied in the past due to its encoding complexity, today, thanks to different optimization techniques[9,10] and to the evolution in hardware and software technologies, the FGS appears a good and feasible solution for multipoint multimedia broadcast systems for the immediate future. Besides, Ref. 11 is an invention of 2006 that discloses methods, devices and systems for effective fine granularity scalability coding and decoding of video data.

A key component of the proposed system is the Topology Manager, which resides into the video source and maintains the tree network topology by deciding the position of each peer within the tree. It receives the uplink bandwidths of all the peers and implements an algorithm to manage both peer arrival and departure events (see Ref. 5 for further details). However, the periodic refresh of the topology may cause frequent oscillations: in real scenarios, for example due to the intensive use of file sharing programs or bandwidth sharing with other users in the same LAN, the bandwidth available to each client of the P2P video distribution platform can strongly oscillate, causing instability in the position assigned by the Topology Manager in the tree. This can produce unacceptable values of delay jitter, that could be not compensated by the playout buffer at destination. As a consequence, due to the presence of this jitter during a video streaming session, on the one hand, buffer underflows provoke video freezing; on the other hand, buffer overflows provoke frame losses. Both cases have to be avoided by giving to the topology manager some extra features when it performs the topology refresh.

The reader notices that in our platform we have preferred the tree topology (see Fig. 1) rather than the mesh[2] because for the considered domain of application saving bandwidth (using tree approach) has priority with respect to keeping low the delay (using

mesh approach). Our choice has been further motivated by the following contras of the mesh topology and the pros of the tree topology.

Below, we report some disadvantages when using mesh topology:

- high chances of redundancy in many of the network connections;
- overall cost of the mesh network is way too high as compared to other network topologies;
- set-up and maintenance of mesh topology is very difficult;
- administration of the network is tough.

On the other hand, some advantages of the tree networks within our domain of application are reported next:

- the expansion of network is possible and easy;
- error detection and correction is easy;
- each segment is provided with dedicated point-to-point wiring to the central hub;
- if one segment is damaged, other segments are not affected.

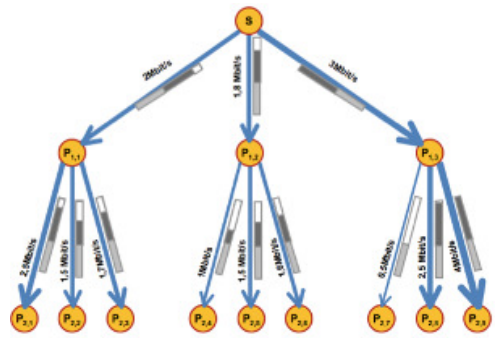

Fig. 1. Example of tree topology.

With all this in mind, the target of this paper is to propose a platform based on hierarchical video streaming to delivery jitter-controlled video in a multipoint fashion for applications with stringent end-to-end delays, where the topology manager performs constrained topology refresh in order to maximize the perceived video quality: given an initial tree topology where peers reside on certain levels, each of them cannot be moved between levels more than M hops. This gives better performances in terms of average PSNR, playout loss, and, therefore, playout frozen time (this is the time during which the video image is frozen because the playout buffer at destination is waiting further frame packets to display).

More in detail, this paper is organized as follows. Section 2 contains the related work. Section 3 describes the system, built on top of Ref. 5, that we propose in this paper. Section 4 shows a case study of the proposed system and a comparison on different parameters of our system with jitter control against a traditional system using a tree topology overlay network. Finally, Section V concludes the paper and indicates the future directions we are headed.

2. Related Work

A lot of work has been done in the past literature on the application of P2P to multipoint multimedia streaming.[16–18] One of the most discussed issues in this context regards the construction and maintenance problems of the overlay network topology.

The majority of the proposed P2P protocols can be broadly classified into the two big categories, according to the used topology: tree-based and mesh-based. The choice of the overlay network topology has a strong impact on overall performance and capacity of reacting to the so-called problem of peer churn.

With the tree-based approach, peers are organized to form a tree-topology overlay network. The source is the root of the tree, data transmission goes from parents to their children until all the peers in the network are reached. Borrowing ideas from IP multicast, tree-based protocols are simple, efficient, and scalable. Systems like Overcast,[19] SpreadIt,[20] PeerCast,[21] NICE,[22] ZigZag,[23] ESM[24] are in this family, only differing for the way they build and maintain the tree. There are two main drawbacks of tree-based streaming systems:

(1) their vulnerability to peer churn. A peer departure will temporarily disrupt video delivery to all peers in the subtree rooted at the departed peer;

(2) all the leaf nodes do not contribute their uploading bandwidth, so degrading the peer bandwidth utilization efficiency.

However, in our scenario, (1) and (2) are not a matter as we are considering a domain where peers are stable; more specific, as far as (2) is concerned, the leaves in our tree structure are most likely to have low-bandwidth to share. For such reasons, the two drawbacks of the tree-based system do not apply in our domain.

SplitStream[25] and CoopNet[26] are prominent examples which use multiple description coding (MDC)[27] to split the stream into different stripes in order to distribute them on several parallel trees (forest topology). However, the overhead of multiple description coding and the complexity of the encoders harm both system efficiency and implementation, and this may raise some concern on the possibility to use efficiently multiple description coding for this kind of application.

With the mesh approach each node maintains a number of partners to exchange with them data availability information.[28,29] Accordingly, a peer actively pushes a received chunk to its neighbors who have not obtained the chunk yet. However, while in tree-based systems a chunk should always be pushed from a peer to all its children peers in the streaming tree, in a mesh-push system there is no clearly defined parent–child relationship. Therefore, a peer might blindly push a chunk to a peer which has already the chunk, and so it might also happen that two peers push the same chunk to the same peer, resulting in bandwidth waste. To address this problem, chunk push schedules need to be carefully planned between neighbors, and the schedules need to be reconstructed upon neighbor arrivals and departures.

Another technique to implement mesh P2P networks is pull based.[30] In a mesh-pull system, video chunks are pulled by a peer from its neighbors who have already obtained the content. To this aim, peers exchange chunk availability using buffer maps periodically, but this adds complexity to the network structure. Other works, for example Ref. 31, proposed to combine pull-based and push-based protocols, in order to take advantage of better resilience to dynamics with a pull-based design, and better delay and stability with push-based protocols.

Since multiple neighbors are maintained at any given moment, mesh-based video streaming systems are highly robust to peer churns for very huge numbers of efficiency unpredictable. Different data packets may traverse different routes to users.

Consequently, users may suffer from video playback quality degradation ranging from low video bit rates, long startup delays, to frequent playback freezes.

In order to decide the technique to be used in our scenario, we have compared the two approaches by simulation, deducing that advantages of the mesh-based ones are not relevant when peer-churn is rare and the number of peers is not too huge. On the contrary, complexity and delay are strongly reduced by using a tree topology.

Another body of work has been done on quality of service (QoS) provided by P2P networks for multipoint multimedia streaming. One of the first attempt was in Ref. 32, where it is shown that using stable peers to form a backbone in the streaming delivery mesh can reduce the transmission delay.

In R2[33] a protocol with random pushing of network-coded blocks is proposed to improve performance as compared with traditional pull-based protocols with or without network coding.

Chunkyspread[34] is an unstructured multiple tree protocol to degrade the transmission delay and to better adapt to the peer dynamics. Very little work focuses on providing delay guarantees to interactive applications using P2P. One of the first work in this direction is Ref. 35, even if it considers delays of about 10 s (not acceptable for

interactive applications), and does not take into account the best-effort nature of the underlying IP network.

To solve this problem, the concept of statistically guaranteed QoS has been introduced in Ref. 23. Bindal *et al.*[36] examine the factors that determine the statistical service guarantee in P2P file sharing applications, such as BitTorrent, while Raghuveer et al.[37] consider how to ensure that each peer gets sufficient bandwidth with a high probability if the system has sufficient overall bandwidth. Kung *et al.*[38] and Xu *et al.*[39] use admission control to determine whether a peer should accept the request of another user to be a neighbor. Different from previous works, our work considers peer admission control (PAC) to ensure that the system is able to statistically guarantee a maximum limit for the end-to-end delay for each peer from the source.

3. System Description

The system we propose in this paper is a live video broadcast platform where a video source distributes a layered encoded video stream to a number of clients in a multipoint fashion. Multipoint communication is achieved by applying a P2P approach, configuring a tree-structured overlay network where the root is the video source, while the other clients are internal nodes or leaves. Video is encoded with a layered encoder, which is a scalar encoder, in order to decouple the encoding process from the time variant behavior of the bandwidth instant-by-instant available in each link of the overlay network. More specifically, we use a two-layer scalable video encoder with fine granularity scalability feature, the MPEG FGS encoder.[7] However, any layered encoding scheme can be used in the platform.

In the following, clients will be also referred to as nodes of the tree, or peers. Let us note that the considered system can be easily extended to a multi-source scenario by using one distribution tree for each source. A tree topology is adopted because it is more suitable for real-time applications with stringent end-to-end delay requirements. The only drawback of a tree topology, network instability due to peer abandons, is not a matter for interactive services where abandons are rare events, mainly limited in cases of client system crashes. In our work we will consider the behavior of the underlying IP network in terms of both bandwidth and delay.TPG

Network topology is updated periodically. The topology refresh period, whose duration expressed in seconds is indicated as ΔT_{TPG}, constitutes an important system parameter. Its choice has a strong impact on the overall system performance, as discussed in Sec. 4.

For each peer p, let us define the so-called *fan-out* parameter as the maximum number of peers that can be attached to p as children. For the sake of simplicity, we will assume

that all the peers have the same fan-out parameter, indicated as F_P. Since the video source may be located on a privileged Internet node, its uplink bandwidth can be greater than the one of the other peers, and therefore its fan-out parameter, in the following indicated as F_S, can reasonably be greater than F_P.

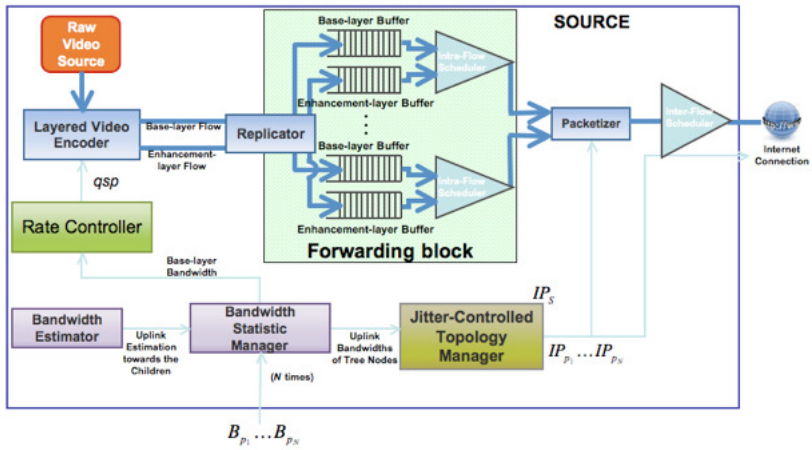

Fig. 2. Architecture block diagram of the video source.

Our system is a generalization of the architecture proposed in Ref. 5. Sections 3.A and 3.B present the architecture of the video source and the generic peer, respectively; Section 3.C describes the algorithm we have defined to manage and update the topology.

A. Source node architecture

As already said so far, topology is organized with a tree structure. For this reason, the video source uses a layered video encoder, which is a scalable encoder producing a fine granularity two-layer stream: the first layer of each produced frame is called Base Layer (BL); it is necessary to decode the frame, and therefore all the peers should receive it. On the contrary, the second layer, called Enhancement layer (EL), is an optional level used to improve quality of the frame. According to the fine granularity layered encoding, the Enhancement layer can be truncated at any point, and the frame quality increases with the amount of EL data each peer receives.

In this way, quality decreases along the tree, from the top to the bottom levels, since it is reduced by bottlenecks. For this reason, high bandwidth peers should be located at the top of the tree; if some low bandwidth peer is located at the top of the tree, the overall quality perceived by each peer is limited by its bandwidth, and therefore the overall average quality will be lower.

The architecture of the video source is shown in Fig. 2. As it can be seen, its core is the *Layered Video Encoder*, which receives a raw video stream as its input and produces the encoded video flow, made up of two separate streams, the BL and the EL ones. A *Replicator* is needed in order to create as many streams (for both Base and Enhancement layers) as the number of the source's children; this number is upper bounded by the source fan-out F_S. Data produced at each frame interval are enqueued into the *Base layer Forwarding Buffer* and *Enhancement Layer Forwarding Buffer*, respectively, in order to avoid possible losses. Then they are sent to the *Intra-Flow Scheduler*, which applies a round-robin strict-priority scheduling algorithm that considers the data in the second buffer (Enhancement layer stream) only in case the first one (Base layer Buffer) is empty and then data are grouped in packets by a *Packetizer*. At most F_S streams end up into the *Inter-Flow Scheduler*, which applies a weighted round robin algorithm to send the streams with an amount of bandwidth proportional to the uplink bandwidth estimated towards each direct children of the source. The amount of bits to be used in the Base layer, and the relative encoding quality, are determined by the *Rate Controller* through the quantizer scale parameter (*qsp*), that is chosen in the range between 1 and 31: the greater the *qsp* value, the poorer the encoding quality.

The Rate Controller obtains the needed information about the bandwidth from the *Bandwidth Statistic Manager*. The latter periodically receives the source uplink bandwidth estimation from the *Bandwidth Estimator* and, at the same time, the uplink bandwidth estimation by all the other peers which are internal nodes in the tree (see B_{p_1}, …, B_{p_N} in Fig. 2 or B_{p_i} in Fig. 3). Finally, the *Jitter-Controlled Topology Manager* decides and maintains the tree network topology by deciding the position of each peer within the tree. It is an extension of the Topology Manager described in Ref. 5. It will be described in detail in Section 2.C.

In order to avoid excessively strong oscillations of both the encoding quality and the system behavior, the bandwidth values are first smoothed with an exponentially-weighted moving average (EWMA) filter with parameter $\beta^{(RC)}$, defined as follows:

$$\hat{B}_n = \beta^{(RC)} \cdot \hat{B}_{n-1} + \left(1-\beta^{(RC)}\right) \cdot B_n \tag{1}$$

where $\hat{B}_{n-1}$ and $\hat{B}_n$ are the filtered bandwidths at the $(n-1)$th and nth update events, B_n is the instantaneous bandwidth at the n-th update event. The Rate Controller filter parameter $\beta^{(RC)}$ ranges between 0 and 1; values of $\beta^{(RC)}$ close to 1 give more importance to the history of the bandwidth process, achieving a process that is less sensitive to high "frequencies" (therefore able to smooth the short-term variations), having slower responses to bandwidth changes; conversely, very low values of $\beta^{(RC)}$ give more importance to recent measures, achieving greater responsiveness to the process

variations. In our implementation we have used a Rate Controller EWMA ($EWMA_{RC}$) filter with $\beta^{(RC)} = 0.8$.

B. Generic client node architecture

Each client node in the overlay network mainly performs three functions:

(1) video play-out;
(2) packet forwarding;
(3) bandwidth estimation.

Its architecture is shown in Fig. 3. The video stream, organized in IP packets, are received by the *Packet Classifier*, which extracts video frame data from the received packets and subdivides them according to their type (Base layer and Enhancement layer) and send them to the *Forwarding block*. At the same time packets are enqueued in the *Playout Buffer* waiting for the time to be given to the local Layered Video Decoder for playback.

In addition to video play-out and forwarding, another important function performed by each peer is the uplink bandwidth estimation towards their children. This task is operated by the *Bandwidth Estimator*, which periodically sends the estimated bandwidth values to the Bandwidth Statistic Manager of the source discussed for the video source diagram of Fig. 2. This part of the system is completely general and any bandwidth estimation algorithm can be plugged in. Its choice goes beyond the purpose of this paper. In addition, sophisticated algorithms to predict the bandwidth in the short or middle term can be applied (see for example Ref. 8).

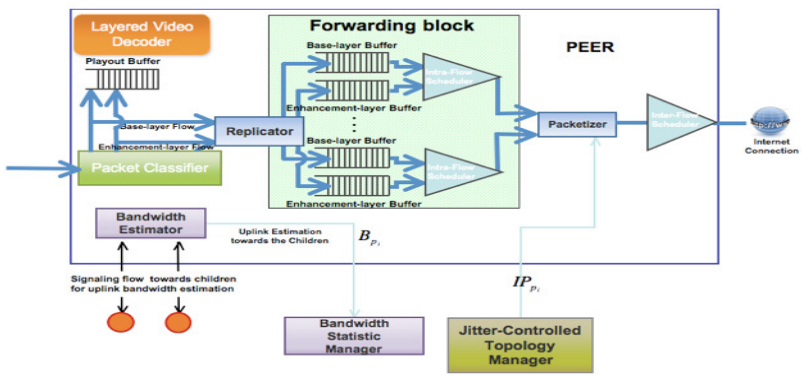

Fig. 3. Architecture block diagram of the generic peer.

C. Jitter-controlled topology manager

The Jitter-Controlled Topology Manager (JCTM) plays an important role for the efficiency of the platform. Its task is to maintain the tree network topology by deciding the position of each peer within the tree. It receives the uplink bandwidths of all the peers from the Bandwidth Statistic Manager, and implements an algorithm to manage both peer arrival and departure events. Of course, the shorter ΔT_{TPG}, the better the topology, as it is more likely that the best peers will always be at the top of the tree. However, changes in the topology structure can be deleterious for video decoding and play-out for some peers; in fact, changing their position in the tree can cause sudden changes in delays, thus increasing the delay jitter as well. If such a delay variation is too high, the Playout Buffer of some peers will not be able to compensate it, and this will cause frame losses at destination. Therefore, it is necessary to keep low this delay variation.

More specifically, the JCTM maintains the tree network topology deciding the position of each peer within the tree every ΔT_{TPG} seconds: it chooses the F_S peers with the highest bandwidth, and connects them as children of the source. Then, for each of these peers, the same operation is repeated, choosing the next F_P peers with the highest uplink bandwidth. This algorithm is run recursively until all the peers get a position in the tree.

To limit the delay jitter discussed above, we have added a constraint to the above topology update algorithm, limiting the maximum jump length, expressed in number of levels, within the tree during a topology refresh. Let i be the generic topology change event, δ_i the time interval between the topology change events i and $i+1$, and T_i the topology during the time interval δ_i. For the generic node k, let $l_k^{(\delta_i)}$ be its level in the tree topology T_i during the interval δ_i.

Indicating with M the maximum length of the jump a peer can do in a single topology update event, the JCTM algorithm must satisfy the following condition:

$$l_k^{(\delta_i)} - M \le l_k^{(\delta_{i+1})} \le l_k^{(\delta_i)} + M \tag{2}$$

In other words, during a topology refresh from topology T_i to topology T_{i+1}, a given peer cannot be moved between levels more than M hops. The parameter M is a system design parameter, whose choice will be discussed in Sec. 4.

Figure 4 shows what happens during the generic ith topology update event. The value t_i on the x-axis indicates the time when the topology starts changing. Let us note that the topology varies each ΔT_{TPG} seconds, thus $t_{i+1} - t_i = \Delta T_{TPG}$. The value ΔT_i is the time during which the old topology T_i and the new topology T_{i+1} coexist together. We refer to this period as the *transient network topology period*. The interval between two

consecutive transient network topology periods will be referred to as *steady network topology period*.

Of course, each peer has a different local transient period, starting when the peer begins receiving packets on the new topology (i.e. from its new parent) and lasting until it has transmitted all the remaining packets to its children on the old topology. This period is highlighted in Fig. 4 with a dark rectangle. New topology packets will be transmitted only after the Forwarding Buffers of the old topology have been emptied. At the end of this transient period the Forwarding Buffers are removed.

4. Results

In this section, we will report the obtained results using our architecture to stream a given video. Section 4A describes the case study we considered for the numerical analysis. Section 4.B shows the results concerning the average duration of topology changes, the loss period duration within the Forwarding Buffer, and the average playout frozen time caused by topology changes for the transmitted video. Then, in Section 4.C we will report the performance in terms of PSNR for the transmitted video using different configurations.

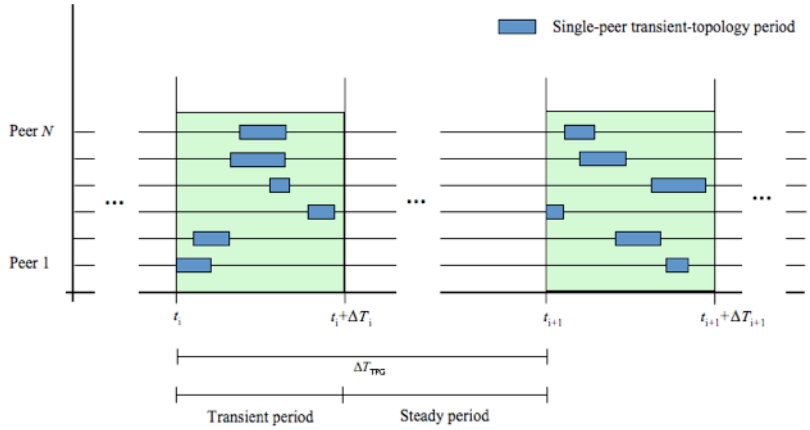

Fig. 4. Transient and steady periods of the network topology.

A. Case study description

We will carry out a steady-state analysis, assuming that the number of peers in the network, hereafter referred to as N, remains constant. In this way our study does not depend on the particular algorithm used to manage the topology structure when peer arrivals or departures occur. Therefore, we focus our attention only on the JCTM job of rearranging the tree according to bandwidth variations following (1). Management of transitory (departures and arrival) peers will be discussed in the Future Work section.

In our implementation we used MPEG-4 FGS encoding: the Base layer is obtained with a classical MPEG-4 encoder using non-scalable coding, whereas the Enhancement layer is coded using a fine-granular scheme. The latter encodes the difference between the original picture and the reconstructed one with the use of bit-plane coding of the DCT coefficients. The encoder has been implemented in Visual C++ using the Intel Integrated Performance Primitives (Intel IPP),[12] an extensive library of multicore-ready, highly optimized software functions for multimedia data processing, and communications applications. Running on a start-level dual-core personal computer equipped with 2 GB of RAM, it is able to encode about 100 fps and decode about 400 fps for CIF video streams. Peers are characterized by different Internet access link performances and different average values of the uplink bandwidth. As previously stated, the downlink bandwidth of each peer is assumed to be much higher than the corresponding uplink bandwidth. Therefore, along this section we will refer to the uplink bandwidth as the bandwidth. We have assumed that the source is a high-bandwidth server with an uplink of 5 Mbit/s. The available uplink bandwidth process of each peer has been generated as in Ref. 5. We have considered a live video stream from the "BBC News channel". The video stream has been encoded with a 176 × 144 QCIF format, at a frame rate of 25 frame/seconds, and using a Group of Pictures (GoP) with a structure given by the pattern IBBPBB. We used ffmpeg[a] to capture the video stream; the obtained video stream has

[a] http://www.ffmpeg.org

been then used by our system for FGS encoding. Our system has been run over Planetlab[b] with 1000 peers. We have run simulations to analyze the behavior of the system when the number of peers exceeded 1000 and did not notice any substantial change within the obtained results with respect to the case of 1000 or fewer peers. Therefore we have decided to not include them here for space constraint.

B. JCTM performance

In this section, we will report a topology analysis of our platform. First, let us analyze the average duration of the transient periods for different values of *M*. Figure 5 shows the results for ΔT_{TPG} = 120 s. The reader notices that the value $M = inf$ represents the case of a traditional system, with no control on the peer jump length during a topology change. It can be observed that, for increasing values of the Forwarding Buffer size, as expected, the average duration of topology changes increases as well. Moreover, the lower the value of M, the lower the average duration of topology changes. This is more evident if we focus, for example, on the case for $M = 1$: during a topology change, peers will be moved at most one level up or down and, therefore, transient packets will arrive to destination sooner and therefore they suffer a lower delay variation.

Figure 6 shows the loss period duration for the Forwarding Buffer. Loss period duration is a very important parameter as the receiver is sensitive to losses if they are too long. Such losses occur in the Forwarding Buffers of the new topology; these losses are due to the fact that Forwarding Buffer of each peer enqueues packets coming from the

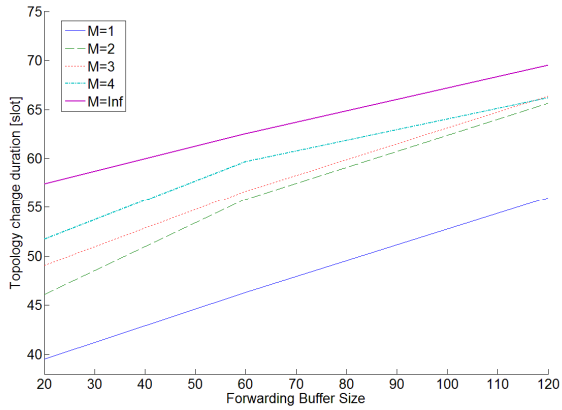

Fig. 5. Average duration of topology changes for ΔT_{TPG}=120s.

[b] http://www.planet-lab.org

new topology, and they are not processed until the packets present in the old topology buffer are not forwarded.

Obviously, the larger the Forwarding Buffers, the higher the number of packets they may contain, and, therefore, more time is needed to process them; it follows that the loss period duration in the new topology buffers increases.

As shown in the figure, our system (for any value of M) outperforms the traditional system ($M = inf$) in terms of loss period duration.

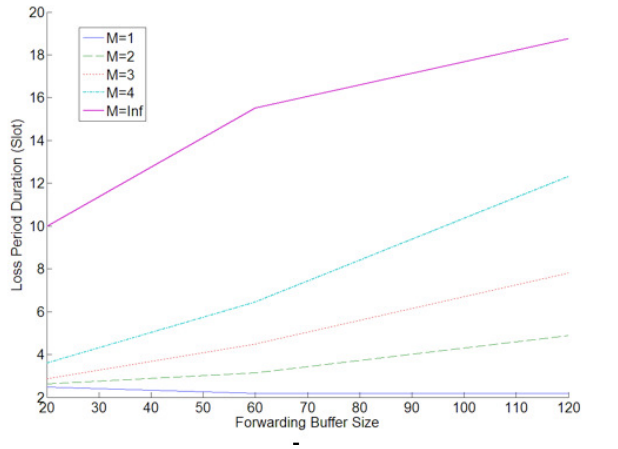

Fig. 6. Loss period duration for the Forwarding Buffers.

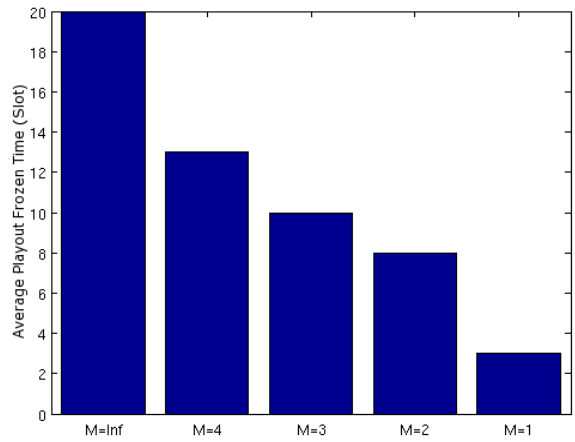

Fig. 7: Average playout frozen time for topology changes.

Finally, Fig. 7 reports the histogram of the playout frozen time averaged over all the topology changes and considering a Forwarding Buffer size equal to 80. The reader may

notice as our system outperforms the traditional system also in terms of playout frozen time, as it is kept much lower.

C. PSNR for the transmitted video sequence

In this section we will first show the numerical results concerning the perceived PSNR varying M in {1, 2, 3, 4, Inf} and ΔT_{TPG} in {120 s, 160 s, 200 s, 240 s}. Let us note that the PSNR perceived by each peer is worsen at each bottleneck, since bottlenecks narrow the bandwidth and reduce the amount of FGS layer received by the descendant peers. Fig. 8 shows such values.

First of all, the reader may notice that, for increasing values of ΔT_{TPG} , the PSNR value gets worse. This happens because higher values of ΔT_{TPG} cause a less frequent update of the topology.

On the other hand, for decreasing values of *M*, the PSNR gets worse. This derives from the fact that a higher number of uplink bandwidth bottlenecks in the overlay network tree topology exist for lower values of *M* according to (1), and this negatively affects the PSNR computation. However, looking at Figs. 6 and 7, the loss period duration and the playout frozen time are much higher as *M* increases.

Then, depending on which scenario we are considering, the following considerations may be expressed:

(i) if the overlay network is not very big or we are transmitting a non-real-time video, and all the participating peers have the same amount of uplink bandwidth, then we may consider a setup with high values of *M* and high values of ΔT_{TPG} , as it will be difficult to incur in packet loss and, beside, we will get higher PSNR values.
(ii) Conversely, if our overlay network is very dynamic and heterogeneous (with a consistent presence of heterogeneous peers in terms of bandwidth), JCTM application becomes necessary, and it must be very stringent, that is, with low values of *M*. The result is that, although overall PSNR is slightly penalized, as shown in Fig. 8, we have a strong improvement of the other performance parameters, in terms of loss period duration and average playout frozen time during topology change periods. Moreover, the choice of ΔT_{TPG} plays a fundamental role in this case, since more frequent topology refreshes are needed to improve perceived PSNR. Of course, we need to be careful with this choice as for low values of ΔT_{TPG} we will have a high number of topology updates, and, consequently, a loss period which happens more frequently (it occurs for each topology update event).

5. Conclusions and Future Work

In this paper we have proposed a platform based on hierarchical video streaming to delivery jitter-controlled layered encoded video in a multipoint fashion for applications with stringent end-to-end delays in order to maximize the perceived video quality and minimize the playout frozen time. As future work, we are already working on the introduction of the Adaptive Media Playout to further reduce potential delays. Moreover, the entire study is based on steady-state, that is, with a constant number of peers. As the bandwidth variations can be significant during churns, we are considering performance of the system in presence of peer churn (in presence of peer departures and arrivals) as a future work.

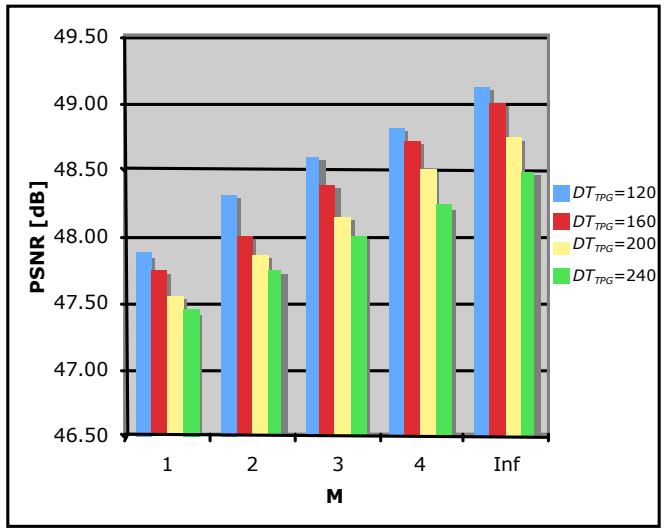

Fig. 8: PSNR comparison between a traditional system without the jitter-controlled condition (*M=Inf*) and the proposed system, for different values of *M*

Acknowledgments

The authors would like to thank anonymous reviewers for useful comments, which have enhanced the quality of the paper.

References

1. X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system", IEEE Transactions on Multimedia, vol. 9, n. 8, Dec. 2007.

2. N. Margharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-based streaming", in Proc. IEEE Infocom 2007, Anchorage, Alasks, USA, May 2007, pp. 1415-1423.F. T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays. Trees. Hypercube, Morgan Kaufmann, 1992.
3. B. Girod, J. Chakareski, M. Kalman, Yi J. Liang, E. Setton, and R. Zhang, "Advances in Network-adaptive Video Streaming", Conf. on Video-Streaming, 2002.
4. Yi J. Liang, E. Setton, and B. Girod, "Channel-Adaptive Video Streaming Using Packet Path Diversity and Rate-Distortion Optimized Reference Picture Selection", IEEE Workshop on Multimedia Signal Processing, 2002.
5. A. Lombardo, D. Reforgiato, and G. Schembra, "P2P and MPEG FGS Encoding: A Good Recipe for Multipoint Video Transmission on the Internet", in International Journal of Digital Multimedia Broadcasting, vol. 2009.
6. W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 3, pp. 301–317, 2001.
7. H. Radha et alii, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," IEEE Trans. on Multimedia, vol. 3, no. 1, pp. 53–68, 2001.
8. W. Li, "Fine granularity scalability in MPEG-4 for streaming video", in Proc. ISCAS 2000, vol 1, Geneva, Switzerland, May 28-31 2000, pp. 299-302
9. B. Krithikaivasan, Y. Zeng, K. Deka, and D. Medhi, "Arch-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic," IEEE/ACM Transactions on Networking, vol. 15, no. 3, 2007.
10. S. Kim, and Y. Ho, "Fine Granular Scalable Video Coding Using Context-Based Binary Arithmetic Coding for Bit-Plane Coding", IEEE Transactions on Circuits and Systems for Video Technology, vol 17, n. 10, 2007, pp. 1301-1310.
11. F. Wu, S. Li and alii, "A Framework for Efficient Progressive Fine Granularity Scalable Video Coding", IEEE Trans. on Circuits and Syst. for Video Tech., vol 11, n. 3, March 2001.
12. Y. K. Wang, "Method, device and system for effective fine granularity scalability (FGS) coding and decoding of video data," International Application N. PCT/IB2006/000631, International Filing Data 22.03.2006, 2006.
13. "Intel(r) integrated performance primitives (intel(r) ipp)," http://www.intel.com/cd/software/products/asmo-na/eng/302910.htm
14. Heiko Schwarz, Detlev Marpe, and Thomas Wiegand:
15. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Scalable Video Coding, Vol. 17, No. 9, pp. 1103-1120, September 2007.
16. Liu Y, Guo Y, Liang C (2008) A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications, No. 1, pp. 18–28
17. Marfia G, Pau G, Di Rico P, Gerla M (2007) P2P Streaming systems: a survey and experiments. 3rd STMicroelectronics STreaming Day (STreaming Day'07), Genoa, Italy, September 14, 2007

18. Sentinelli A, Celetto L, Lefol D, Palazzi C, Pau G, Zahariadis T, Jari A (2009) "A Survey on P2P Overlay Streaming Clients", IOS press "Towards the Future Internet—A European Research Perspective," ISBN 978-1-60750-007-0 pp. 273–282
19. Jannotti J, Gifford DK, Johnson KL, Kaashoek MF, JWO Jr. (2000) Overcast: reliable multicasting with an overlay network. In Proc. Operating Systems Design and Implementation, pp. 292–301, October 2000
20. Deshpande H, Bawa M, Garcia-Molina H (2001) Streaming live media over a peer-to-peer network. Technical Report of Database Group, CS-2001-31, Stanford University
21. Zhang J, Liu L, Ramaswamy L (2007) PeerCast: churn-resilient end system multicast on heterogeneous overlay networks. Journal of Network and Computer Applications (JNCA), Elsevier
22. Banerjee S, Bhattacharjee B, Kommareddy C (2002) Scalable Application Layer Multicast. Proc. of ACM Sigcomm 2002, Pittsburgh, Pennsylvania, August 2002
23. Tran DA, Hua KA, Do T (2003) ZIGZAG: an efficient peer-to-peer scheme for media streaming. Proc. Of IEEE INFOCOM 2003, pp. 1283–1292
24. Chu Y-H, Rao SG, Zhang H (2000) A case for end system multicast. In Proceedings of ACM SIGMETRICS
25. Castro M, Druschel P, Kermarrec A-M, Nandi A, Rowstron A, Singh A (2003) SplitStream: high- bandwidth content distribution in a cooperative environment. In: Kaashoek MF, Stoica I (eds) IPTPS 2003. LNCS, vol. 2735. Springer, Heidelberg
26. Padmanabhan VN, Sripanidkulchai K (2002) The case for cooperative network-ing. In: Druschel P, Kaashoek MF, Rowstron A (eds) IPTPS 2002. LNCS, vol. 2429. Springer, Heidelberg, pp 178–190
27. Goyal VK (2001) Multiple description coding: compression meets the network. Signal Process Mag 18 (5):74–93
28. Bonald T, Massoulié L, Mathieu F, Perino D, Twigg A (2008) Epidemic live streaming: optimal performance trade-offs. In Proc. SIGMETRICS 2008, Annapolis, Maryland, USA, pp. 325–336, June 2–6, 2008
29. Couto da Silva AP, Leonardi E, Mellia M, Meo M (2008) A bandwidth-aware scheduling strategy for P2P-TV systems. In Proc. IEEE P2P'08, Aachen, DE, September 2008
30. Zhang X, Liu J, Li B, Yum TP (2005) CoolStreaming/DONet: a data-driven overlay network for efficient live media streaming. infocom
31. Zhang M, Xiong Y, Zhang Q, Yang S (2005) A peer-to-peer network for live media streaming: using a push-pull approach. In Proc. Of ACM Multimedia 2005, November 2005
32. Wang F et al (2008) Stable peers: existence, importance, and application in peer-to-peer live video streaming. In Proc. of IEEE INFOCOM 2008, April 2008
33. Wang M, Li B (2007) R2: random push with random network coding in live peer-to-peer streaming. IEEE JSAC, Dec 2007
34. Venkatraman V, Francis P (2005) ChunkySpread overlay multicast. In Proc. 2nd Symposium on Networked Systems Design and Implementation, May 2005

35. Zhang M, Sun L, Xi X, Yang S (2008) iGridMedia: providing delay-guaranteed peer-to-peer live streaming service on internet. In Proceedings of GLOBECOM'2008. pp. 1741–17
36. Bindal R, Cao P (2006) Can self-organizing P2P file distribution provide QoS guarantees. ACM Operating Systems Review 40(3):22–30
37. Raghuveer A, Dong Y, Du D (2007) On providing reliability guarantees in live video streaming with collaborative clients. In Proceedings of MMCN (San Jose, CA, January 2007)
38. Kung H, Wu C (2003) Differentiated admission for peer-to-peer systems: incentivizing peers to contribute their resources. In Proceedings of Workshop on Economics of Peer-to-Peer Systems (Berkely, CA, June 2003)
39. Xu D, Hefeeda M, Hambrusch S, Bhargava B (2002) On peer-to-peer media streaming. In Proceedings of IEEE ICDCS (Austria, July 2002)