Push–pull techniques in peer-to-peer video streaming systems with tree/forest topology

Paolo Giacomazzi, Alessandro Poli
Politecnico di Milano, Dept. of Electronics and Information
Piazza Leonardo Da Vinci 32, 20133 Milano, ITALY
E-mail: {giacomaz, poli}@elet.polimi.it

***Abstract*—Peer-to-peer video streaming systems are overlay networks used to distribute, among other types of content, *live* video content to large sets of users by relying on computing and network resources directly provided by users that are receiving video streaming services. Peer-to-peer video streaming systems with tree or forest topology are typically *push-based*, since the video content is provided by parent peers with no need for periodical requests. In this paper we analyze the impact of two complementing *pull-based* mechanisms aiming at improving the overall performance of the overlay network. Results show that the proposed hybrid push-pull approaches can be beneficial when the stability of the system is low, i.e., the average permanence time of peers is short.**

Peer-to-peer; video streaming; tree; retransmission; backup parents; performance analysis

I. INTRODUCTION

PEER-TO-PEER systems are an effective way for content providers to distribute media content to a large set of users, with limited investments for network infrastructures. These systems rely on the provisioning of network and computing resources by users that are receiving video streaming services. These systems are self-scalable, since an increase in the number of users is compensated by an additional availability of resources. However, performance can be greatly affected by the uncontrollable behavior of peers, who can disconnect without notification, negatively impacting on the quality of the video stream received by other users. Users, referred to also as *peers*, receive the video stream from other peers and forward it to one or multiple peers. This *multicast-like* paradigm can be achieved by creating an overlay network through which the content is exchanged. However, because of the high instability of peers and their unpredictable behavior, the quality of the media stream received by customers could be very low.

In the literature, the performance analysis of peer-to-peer video streaming systems is carried out by means of three techniques: (1) *trace analysis* of working systems or the deployment and study of prototypal systems on specific *test beds*, (2) *analytical studies*, and (3) *simulation*.

The largest part of studies belongs to category (1). Paper [1] proposes a measurement study of the popular PPLive system [2], carried out by means of a dedicated crawler. The cited work studies users' behavior, peers data exchange and playback delays. The authors of [3], using a combination of analysis and real traces of CoolStreaming [4], study how buffering techniques are used to cope with system dynamics and heterogeneity. Magellan [5] is a project launched with the objective of gaining in-depth insights on P2P streaming characteristics. The authors used 120 GB of traces from a commercial system to explore the behavior of some topological properties over the time. Paper [6] provides a survey and a set of experiments on popular P2P video streaming systems, measuring performance indexes, such as the ratio of lost frames and the playback delay. The authors of [7] developed a framework to analyze two popular P2P video streaming systems; resource usage, locality, and stability of data distribution is then studied. In [8], an analysis platform for P2P video streaming is presented, taking into account the interactions between peers and the underlying network; the platform allows to connect a real P2P video client (purposely modified) to a network simulator, and to study the rate of loss frames for each peer.

Paper [9] belongs to category (2); the authors provide a stochastic model, used to compare different downloading strategies based on two performance metrics: probability of continuous playback, and startup latency. In [10] an analytical model of a real-time P2P video streaming system is proposed, in order to estimate some performance parameters, such as average delay, and service provider revenue, as function of the nodes preemption probabilities.

Simulation (3) is used in [11], for a comparative study between the tree-based and the mesh-based approaches. The authors of the referred work study the effects of the bandwidth of connections, peer degree, bandwidth heterogeneity, group size, and churn. Paper [12] is another analysis, performed by simulation, of a reference P2P video streaming system with Multiple Description Coding; it provides an evaluation of video quality.

Ordinary peer-to-peer video streaming systems with tree and forest (multiple trees) topology are typically *push-based*, since the transmission of data is guided by the senders and

978-1-4244-7286-4/10/$26.00 ©2010 IEEE

does not require any request by the receivers. In this paper, we analyze the impact of the introduction of two forms of *pull* techniques in a push-based system: the *retransmission* of packets lost during the disconnections, and the availability of *backup parents*.

In the literature, there are several studies on hybrid push-pull techniques, for example [14], [15] and [16], but they focus on pull-push mechanisms built on top of an original pull-based content delivery approach, while in our work we examine simple improvements applied to tree and forest topologies, without modifying the original delivery approach. This allows us to better identify the marginal contribution of these techniques to the overall quality of the media distribution. A work focused on backup parents is [17], however it does not investigate the conditions under which this technique can be beneficial.

We carry out our study by means of a fine-grained simulative modeling of the peer-to-peer video streaming system that is fully disclosed in this paper. To the best of our knowledge, our simulative model of the system is significantly more accurate than similar extant models.

II. The Base Reference Model

In this work, we focus on tree-based peer-to-peer video streaming systems. Our model is inspired to VidTorrent [13], a tree-based peer-to-peer video streaming system developed at the Massachusetts Institute of Technology. However, our model is more general and it accounts for peer-to-peer video streaming systems with the following properties: (1) a unique content distribution source is responsible for the provisioning of the video stream to the whole system, (2) the structure of the distribution is a tree or a forest, (3) at the application level, in the overlay peer-to-peer network, the content is organized into chunks of video frames referred to as *segments*, (4) a single *frame* can be split into a fixed number (≥1) of *sub-frames* of variable length, (5) users can join and leave the peer-to-peer system dynamically, even during the distribution of a video.

In the following sections the model of the base reference system is explained in detail. Section III will describe the modification to this model introduced in order to simulate the behavior of the system when the proposed pull-based techniques are used.

A. The Video Stream

The video stream provided by the source is a sequence of m ordered frames. We identify a single *frame* f_i by its frame number $i = 1, 2, \ldots, m$. For each frame we know the start time $f_i.start$ and the end time $f_i.end$ in seconds, identifying the time interval covered by the frame with respect to the entire video stream. Every frame is split into n *sub-frames*, where a sub-frame represents a part of the whole frame, such as, for example, a specific layer in a layered coding, or a single description in a Multiple Description Coding (MDC)[1]. A sub-frame sf_{ij} is identified by the frame number i and the sub-frame offset $j = 1, 2, \ldots, n$. When coping with single description/single layer coding, a frame has just one sub-frame ($n = 1$). For each sub-frame we know the length $sf_{ij}.length$ in bytes. Sub-frames can have either variable or fixed size.

B. Segments

At the application layer, chunks of k sub-frames are organized into segments. A segment s_i is assembled by grouping sub-frames having the same offset in consecutive frames (see Fig. 1). For example, when $n = 4$ and $k = 3$, the first segment of the video stream s_1 is made of the sub-frames sf_{11}, sf_{21}, and sf_{31}, while s_2 is made of the sub-frames sf_{12}, sf_{22}, and sf_{32}.

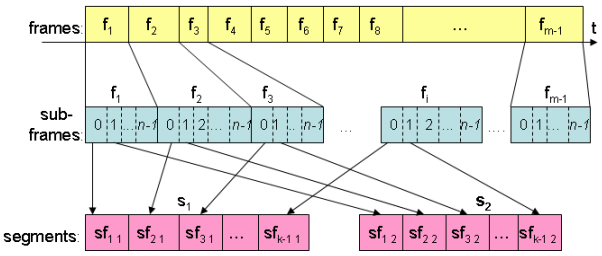

Fig. 1. Frames, sub-frames and segments.

C. Source and Trees

The video content is distributed among all peers through a set of q independent trees. The source provides the video stream and is placed at the root of each tree. It sequentially sends segments to its children at the rate determined by frame start and end times. The source has a limited amount of bandwidth Sup, measured in bit/s.

The number of trees q is a multiple of the number of sub-frames per frame n, such that only the segments composed by the sub-frames with the same j-th sub-frame offset are forwarded in the same tree. A variable number d of trees is allowed to transport segments with the same sub-frame offset (*tree diversity* property). The total amount of trees is thus $q = d \cdot n$. Every segment is sent through the appropriate tree, alternating among the d available trees.

D. Trees and Peers

A client in the peer-to-peer video streaming system is called *peer*. A peer, identified by p_i, in order to receive the video content, must be a node of the trees carrying the content. A peer is not required to be part of all trees. For example, it

[1] In a Multiple description coding (MDC) a *description* is a sub-stream of a single media stream. The segments of each description are routed over multiple independent overlay trees.

could be a node of the d trees transporting the segments made up of sub-frames with the same single sub-frame offset. With a Multiple Description Coding this would translate into the reception of a single description, causing the display of a degraded version of the video stream.

All peers, for each tree they are part of, receive segments from their parents and then send them to their children. A peer can be placed in different positions in different trees, and different trees can have a different topology.

E. Peers Operations

Each peer is responsible for the distribution of the media. It operates both at the overlay (application) and the underlay (network) levels. The access bandwidth of peer p_i, expressed in bit/s, is referred to as $p_i.Cup$ and $p_i.Cdown$, representing the upload and the download capacity of the access network, respectively. The peer is provided with a transmission buffer (in the upload direction) and a reception buffer (in the download direction).

Each peer performs the following actions (see Fig. 2):

1. The download queue, where the packets from parent peers are received, is emptied at a rate determined by $p_i.Cdown$.
2. All sub-frames received from the download queue are stored in an overlay buffer, named playout buffer, that reassembles the stream, according to frame numbers and sub-frame offsets.
3. As soon as all the sub-frames forming a segment are received, the segment is divided into packets and sent multiple times to all the children peers, through the appropriate trees. These packets are transmitted through the upload link.
4. The frames stored in the playout buffer are sequentially extracted by the client's player at the rate determined by the video stream.

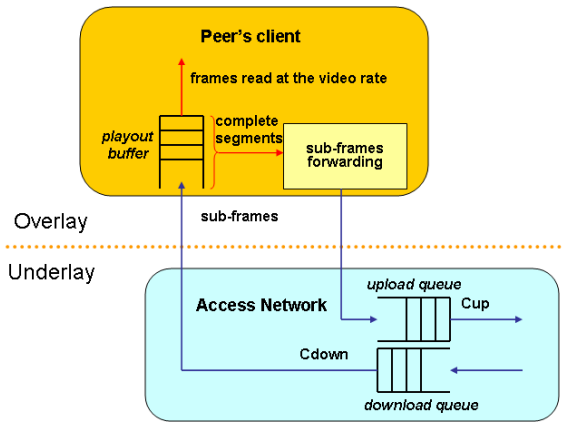

Fig. 2. Peer client and access network.

F. Playout Buffer

The playout buffer is responsible for the re-assembly of the video stream (segments are carried by multiple packets in the underlay network). The playout buffer has a finite length, $PBlength$, measured in segments. When a sub-frame is received, it is placed in the correct position in the playout buffer. When a complete segment is reassembled in the playout buffer, it is sent to the children, as described in the previous section. When all the frames of a segment have been read by the player, a position in the playout buffer is freed.

A peer starts playing the video stream as soon as a playback threshold $PBTh$, measured in seconds, is reached. The reaching of the threshold is computed independently for every sub-frame offset. The playback starts as soon as, for at least one sub-frame offset (corresponding to a layer or a description in a layered or in a multiple description coding), the threshold has been exceeded.

G. Join

When a new peer wants to receive the video stream, it must become part of one or multiple distribution trees. In order to play the video stream, every peer must join at least the d trees transporting the segments composed of sub-frames with the same sub-frame offset. In the following, the *join* procedure is described.

The *join* procedure connects the new peers at the highest levels in the trees, without any optimization based on peers' access network capacity. In details, for peer p_i:

1. Depending on the free download bandwidth (equal to $p_i.Cdown$ if the peer is not already receiving any other sub-stream), the maximum number of sub-frames per frame to be received is computed.
2. The sub-frame offsets are chosen randomly.
3. For every chosen sub-frame offset, the peer selects a parent in all the d trees for that offset (a parent can be either another peer or the source). For each tree, the parent is randomly chosen among the peers at the highest level in the tree (nearer to the source) with a sufficient amount of free upload bandwidth.
4. After a time interval t_{join}, measured in seconds, the new node starts receiving the sub-streams from its new parents. This interval models the time required for the identification and the selection of a parent.

H. Leave

A peer can leave the system unpredictably and without notification. Whenever this happens, its children – and, iteratively, all their grandchildren in the same tree – cease to receive the sub-stream. In the following, the *leave* procedure is described.

When a peer leaves the system, the orphan peers start the join procedure for each tree they have been disconnected from, and start receiving the sub-stream from their new parents after a time interval t_{rejoin_leave}. This time interval can represent, for example, the time required by a keep-alive failure detection mechanism for the identification of the leave of a

parent and the subsequent search for a new candidate parent. The join procedure for an orphan peer is identical to the procedure followed by a new peer, as described above, but is related exclusively to the sub-frame offset the orphan peer was receiving from its dead parent. Only direct children of the dead peer try to rejoin the trees in new positions, while all the isolated trees move together with their ancestors.

III. PUSH-PULL TECHNIQUES

This section presents the modification introduced to the model presented in Section II, in order to provide two interesting *pull* mechanisms (also referred to as *data-driven*) aiming at improving the overall performance of a strictly push-based tree topology.

A. Retransmission Technique

Whenever a parent leaves the system, for all the duration of the time interval t_{rejoin_leave}, the orphan peer looses the segments that are transmitted in the distribution tree. These segments are lost forever, and there is no way to retrieve them at a later time, since the tree and forest topologies are strictly push-based. We introduce a pull mechanism that allows to explicitly requesting lost segments, as soon as the disconnected peer rejoins the system and connects to a new parent.

The new *join* procedure, initiated by orphan peers, includes the following additional steps:

1. When a peer connects to a new parent, it sends the segment number i of the last received segment s_i for the specific sub-frame offset j (i.e. a description, for a multiple description coding).
2. After a time interval t_{join}, the new parent starts sending the sub-stream to its new child, i.e. starting from the last segment received by the parent.
3. Concurrently, the new parent identifies in its playout buffer the segment s_i requested by the new child. If not available it identifies the first segment for the specific sub-frame offset j in the playout buffer having the smaller segment number higher than the segment number i.
4. The parent sends to the new child all the segments in its playout buffer, for the specific sub-frame offset j, spanning from the identified segment to the last received segment. This operation is referred to as *retransmission*, since it is related to the transmission of segments already forwarded to children in the past.

In order to avoid congestion of access links, and consequently affecting the quality of the ordinary stream, retransmitted segments are inserted in a *low priority queue*, which is emptied at the rate Cup – in the same way as the ordinary *upload queue* – when no packets are in the *upload queue*. This expedient has been proved to offer better overall performances in all the simulated scenarios.

For the same reason, we also introduced a maximum fraction of the bandwidth used by the ordinary streams. The number of children a peer can accept is consequently computed against a smaller amount of available upload bandwidth, e.g. the 90%, so that an amount of free capacity can be exploited on request by the additional flows started by the *retransmissions*, reducing the interferences with standard flows. The additional parameter $cap\_threshold$ (equal to 0.9 in the example above) is consequently introduced in the model, and step 3 of the *join* procedure (see Section II.G) is modified, so that the amount of free upload bandwidth for peer p_i is computed considering the total bandwidth value $cap\_threshold \cdot p_i.Cup$.

B. Backup Parents

The second technique investigated in this paper is the creation of a virtual network of *backup parents*. Each peer, in addition to the standard parent keeps a link to a different parent for every sub-frame offset/description. Backup parents are not additional peers, they are standard peers with additional relationships with other children. Backup parents do not transmit anything to their additional children, unless they receive an explicit request. The availability of backup parents allows to rapidly responding to unpredicted leaves of ordinary parents. Whenever a peer discovers the death of one of its parents, it can immediately rely on a backup parent, avoiding the packet losses during the time interval t_{rejoin_leave}. In a strictly push-based system, throughout this time interval the transmitted segments are lost forever. The leave of a parent can not be immediately detected, and a time interval t_{req_backup} is also considered in our reference system. t_{req_backup} can be smaller than t_{rejoin_leave}, and consequently provide benefits to the system, since there is no need of searching a new parent peer, because the backup parent is always ready to transmit. The virtual network made up by the latent backup parents also evolves, since backup parents leave the system and new backup parents must be identified, in the same way as ordinary parents. The additional operations put in place for the management of backup parents, are concurrently executed when the join and leave procedures for ordinary peers are performed.

In the following, the additional steps executed during the join of a new peer to the system are outlined. These operations allow connecting to a *backup parent*, randomly chosen among all peers (excluding the ordinary parent of the joining peer).

In order to avoid congestion of access links, all the segments directed to *backup children* (peers connected to a given backup parent) are inserted in a *low priority queue*, which is emptied at the rate Cup – in the same way as the ordinary *upload queue* – when no packets are in the *upload queue*.

1) Join to a Backup Parent

When a new peer wants to receive the video stream, it must become part of one or multiple distribution trees. In order to play the video stream, every peer must join at least the d trees transporting the segments composed of sub-frames with the same sub-frame offset. Moreover, the peer must connect to a *backup parent* for each tree it belongs to.

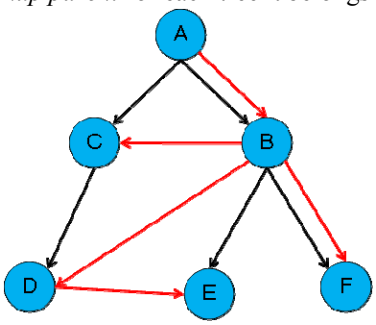

Fig. 3. Standard tree (in black), and links between peers and their backup parents (in red).

In details, for peer p_i:

1. For every sub-frame offset chosen by the *join* procedure, the peer selects a backup parent in all the d trees for that offset (a parent can be either another peer or the source). For each tree, the parent is randomly chosen among all the peers, excluding the ordinary parent already identified by the *join* procedure.
2. After a time interval t_{join}, the new backup peer is ready to accept requests of transmission from its children.

2) Leave

The new *leave* procedure is organized as follows:

1. When a peer leaves the system, orphan peers send a special packet to their backup peers, asking for the transmission of the interrupted sub-streams.
2. Backup peers, after a time interval t_{req_backup}, start sending the requested flow to orphan children.
3. Orphan peers start the join procedure to ordinary parents for each tree they have been disconnected from, and start receiving the sub-stream from their new parents after a time interval t_{rejoin_leave}.
4. If the dead peer was acting as a backup peer, also its backup children start the join to backup peers, for each backup tree they have been disconnected from.
5. Orphan peers send a request of interruption of the transmission to their backup parents, and the flow continues to be received from the new ordinary parents.

IV. Performance Analysis

A. Parameters and Indexes

In this section, we study the impact of three system parameters on the performance of the peer-to-peer video streaming system, when the *retransmission technique* and/or the *backup parents* are used. The presented system parameters have been selected because their values affect the benefits provided by the implementation of the *retransmission technique* and the *backup parents*.

The selected performance parameters are:

1. *average permanence time of peers* in the system, $1/\mu$, measured from peer first join to its leave;
2. *capacity threshold* $cap\_threshold$, the maximum share of the total upload capacity that can be used for ordinary flows towards children;
3. *request time to backup peers* t_{req_backup}, required by a peer to detect the death of a parent, and requesting and start receiving a sub-stream from a backup peer.

We measure system performance by means of the following indexes:

1. *playback delay*, defined as the time elapsing from the instant in which the source provides the content to the instant in which a client reads it from the peers playout buffer;
2. *received frames and sub-frames ratios* for each peer, measured by considering the presence or absence of sub-frames in the playout buffer at their playback time.

B. Simulation Parameters

We have carried out an extensive simulation process by considering a peer-to-peer video streaming system fed with a real video trace of a soccer match with a duration of 36 minutes, coded with a Multiple Description Coding (MDC) with 9 descriptions per frame ($n = 9$). The average rate of the video stream is 943.213 kbps, and every frame has a fixed duration of 33.3 ms. Each segment comprises $k = 20$ sub-frames. A single description is sent through one tree ($d = 1$), such that a total number of $q = 9$ trees/sub-streams are used. We have selected the source bandwidth Sup in such a way that it can provide up to 45 sub-streams (i.e. up to 5 complete streams) simultaneously. The playout buffer length has been set equal to 133.3 s ($PBlength = 1800$), and we have used a playback threshold $PBTh$ of 3.33 s. The join time t_{join} has been considered equal to 500 ms, and the rejoin time t_{rejoin_leave} equal to 100 s. When not otherwise explicitly notified, the average number N of simultaneously active peers has been considered 100, no rewarding technique has been used, the average permanence time of peers has been considered equal to 5 minutes, capacity threshold has been set to 100%, and request time to backup peers to 5 s.

The upload and download access bandwidth ($p_i.Cup$ and $p_i.Cdown$) for joining peers, have been set according to the following probability distribution:

1. 50% – $Cdown = 7$ Mbps , $Cup = 1$ Mbps ;
2. 30% – $Cdown = 20$ Mbps , $Cup = 1$ Mbps ;

3. 10% – $Cdown = 8$ Mbps , $Cup = 1$ Mbps ;
4. 10% – $Cdown = 10$ Mbps , $Cup = 10$ Mbps .

In order to evaluate the benefits that can be achieved by the utilization of rewarding techniques of peers, simulations have been carried out in different scenarios, considering the enabling of the *retransmission technique* and/or the *backup parents*.

C. Results

In this section, the performance indexes presented in Section IV.A are used to compare the behavior of the system by using the *retransmission technique* and/or the *backup parents*, presented in Sections III.A and III.B, as the average permanence time of peers, capacity threshold, and request time to backup peers vary. The reported numerical values are averages on all the peers observed in the system, and have been obtained with a grand average over a set of independent simulations using different seeds for random number generation. The number of simulations for each point is variable and it has been chosen in such a way that the 95%-confidence intervals are smaller than 10% of the average values.

1) Peer Duration

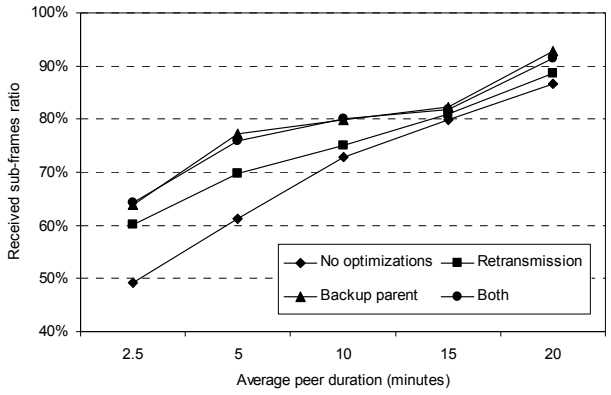

Fig. 4. Ratio of received sub-frames, with varying average permanence time of peers, without optimizations, with the retransmission technique, with backup parents, and with both.

Fig. 4 shows the performance improvement due to the retransmission technique. Benefits are significant, in the order of 10% when the duration of peers is short (2.5 minutes), while decrease when the system is more stable. With average peer duration of 20 minutes, improvements are below 2%, since the retransmission technique acts in response to leaves of peers. The same holds with the use of backup parents and benefits are higher when the system is less stable; in our scenario, improvements are in the order of 15% when the duration of peers is 2.5 minutes. The joint use of the retransmission technique and backup parents, brings to results similar to those obtained with the use of the backup parents only. In the selected simulation scenarios, the backup parents technique is more effective than the retransmission technique.

2) Capacity Threshold

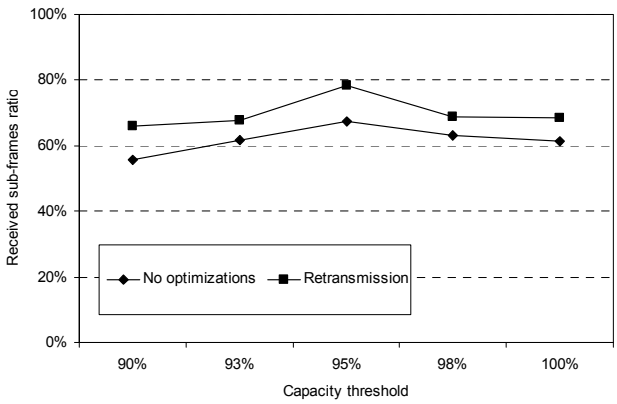

Fig. 5. Ratio of received sub-frames, with varying capacity threshold.

Fig. 5 shows the amount of received sub-frames with varying *capacity threshold* $cap\_threshold$. The capacity threshold represents the maximum share of the total upload capacity that can be used for ordinary flows towards children. The spare bandwidth is useful for reducing the congestion of upload links, and can be exploited by the retransmission technique for sending requested segments. Results shows that best performances can be achieved when 95% of overall upload bandwidth is used for ordinary flows. Lower values bring to inefficient use of resources, while higher values cause congestion of access links.

3) Request Time to Backup Peers

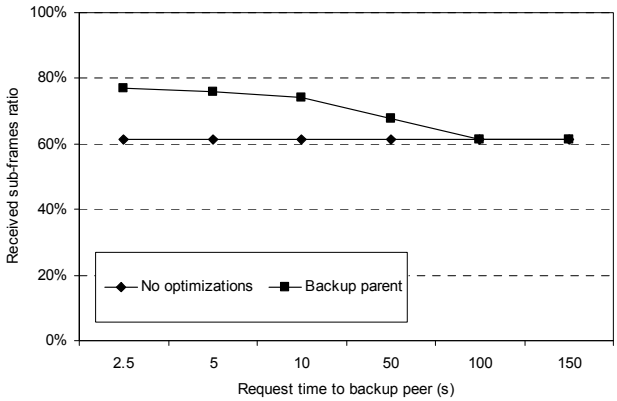

Fig. 6. Ratio of received sub-frames, with varying request time to backup peers.

The use of backup parents is beneficial only if the time to connect to backup parents is smaller than the time typically requested to discover a new parent and connect to it (t_{rejoin_leave}). This condition can be easily satisfied, since a backup parent is always ready to transmit and replace the ordinary parent. In Fig. 6 we plot the *request time to backup peers* t_{req_backup}, i.e. the time required by a peer to detect the

death of a parent, and requesting and start receiving a sub-stream from a backup peer. Results prove that benefits are higher when response time of backup peers is smaller, while when the *request time to backup peers* exceeds the *rejoin time* no benefits can be registered.

We observed that, in order to provide benefits to the peer-to-peer video streaming system through the retransmission technique, a minimum amount of video must be buffered in order to send lost segments to rejoining peers. Performances are maximized when playout buffer length *PBlength* is such that the amount of cached video in seconds is larger than t_{rejoin_leave}.

For the remaining parameters (including the *average number of peers* N), we discovered that their impact on performances does not vary when the retransmission technique or the backup parents techniques are employed.

V. Conclusions

In this work, we have studied the benefits of two forms of optimization, to retrieve lost video segments in peer-to-peer video streaming systems. The retransmission technique is a pull mechanism that enables the retrieval of lost segments as soon as a disconnected peer rejoins the system and connects to a new parent. The backup parents technique employs additional parents used by peers when standard parents are not available. These two techniques allow mitigating the performance degradation that can be observed when the average peer's duration is short.

We discovered that the backup parents technique is beneficial only if the time to connect to backup parents is smaller than the time needed to discover a new parent and connect to it. With the retransmission technique, a minimum amount of video must be buffered in order to send lost segments to rejoining peers. We also observed that peer-to-peer video streaming systems with tree/forest topology offer better performances when a small share of the upload capacity (around 5%) is kept free for retransmissions and not used for the creation of additional links towards children.

References

[1] X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System", *IEEE Transactions on Multimedia*, Dec. 2007, vol. 9(8), pp. 1672 – 1687.

[2] PPLive, http://www.pplive.com/.

[3] S. Xie, G.Y. Keung, B. Li, "A measurement of a large-scale peer-to-peer live video streaming system", *Packet Video 2007*, pp. 153-162, Nov. 2007.

[4] X. Zhang, J. Liu, B. Li, T. S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming", *IEEE Infocom*, 2005.

[5] C. Wu, B. Li, S. Zhao, "Magellan: Charting Large-Scale Peer-to-Peer Live Streaming Topologies", *ICDCS '07, 27th International Conference on Distributed Computing Systems*, pp. 62, June 2007.

[6] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, S. Tewari, "Will IPTV ride the peer-to-peer stream? [Peer-to-Peer Multimedia Streaming]", *IEEE Communications Magazine*, vol.45(6), pp. 86-92, June 2007.

[7] S. Ali, A. Mathur, H. Zhang, "Measurement of commercial peer-to-peer live video streaming", *First Workshop on Recent Advances in Peer-to-Peer Streaming*, August 2006.

[8] M. Barbera, A.G. Busà, A. Lombardo, G. Schembra, "CLAPS: A Cross-Layer Analysis Platform for P2P Video Streaming", *ICC '07. IEEE International Conference on Communications*, pp. 50-56, June 2007.

[9] Y. Zhou, D.M. Chiu, J.C.S. Lui, "A Simple Model for Analyzing P2P Streaming Protocols", ICNP 2007, *IEEE International Conference on Network Protocols*, pp. 226-235, Oct. 2007.

[10] G. Incarbone, G. Schembra, "Peer admission control in a real-time P2P video distribution platform: definition and performance evaluation", *Packet Video 2007*, pp. 163-172, Nov. 2007.

[11] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-tree: A comparative study of live P2P streaming approaches", *Proc. of IEEE INFOCOM'07*, May 2007.

[12] I. Lee, Y. He, L. Guan, "Centralized P2P Streaming with MDC", *IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1-4, Oct. 2005.

[13] I. Mirkin, "Reliable Real-time Stream Distribution Using an Internet Multicast Overlay," http://viral.media.mit.edu/index.php?page=vidtorrent, 2006.

[14] Susu Xie, Bo Li, G.Y. Keung, and Xinyan Zhang, "Coolstreaming: Design, Theory, and Practice," *Multimedia, IEEE Transactions on*, vol.9(8), pp.1661-1671, Dec. 2007.

[15] Li Zhao, at al., "Gridmedia: A Practical Peer-to-Peer Based Live Video Streaming System," *Multimedia Signal Processing, 2005 IEEE 7th Workshop on* , pp.1-4, Nov. 2 2005.

[16] U. Abbasi, M. Mushtaq, T. Ahmed, "Delivering scalable video coding using P2P Small-World based push-pull mechanism," *Information Infrastructure Symposium, 2009. GIIS '09. Global*, pp.1-7, June 2009.

[17] Y.-T.H. Li, Dongni Ren, S.-H.G. Chan, A.C. Begen, "Low-delay mesh with peer churns for peer-to-peer streaming," *Multimedia and Expo, 2009, ICME 2009, IEEE International Conference on*, pp.1546-1547, July 2009.