Performance analysis of optimization 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 network and computing resources directly provided by users that are receiving the video stream. In this paper, we analyze the impact of two optimization techniques that can be adopted in peer-to-peer video streaming systems, with tree or forest topology, to cope with the negative effects of user's leaves. We carry out a performance analysis of these systems, analyzing their sensitivity to the most critical system parameters, when *nearly-permanent* nodes, i.e., peers with a smaller-than-average leave rate, are used to optimize the overlay topology.**

Peer-to-peer; video streaming; tree; optimization techniques; permanent nodes; 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.

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*.

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

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] develope 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.

In this paper, we analyze the performance of peer-to-peer

video streaming systems with tree and forest (multiple trees) topology, when *nearly-permanent* peers (peer with a small leave rate) are used to optimize the system's overlay topology. We analyze the conditions under which this optimization 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 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 reference system is explained in detail.

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}.

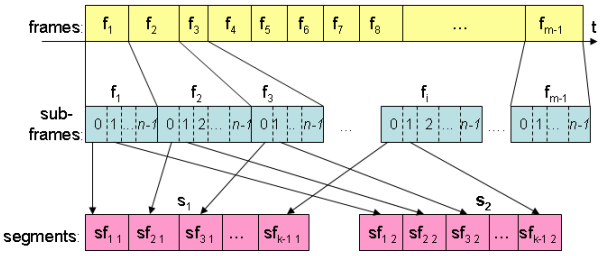

Fig. 1. Frames, sub-frames and segments.

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 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,

[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.

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.

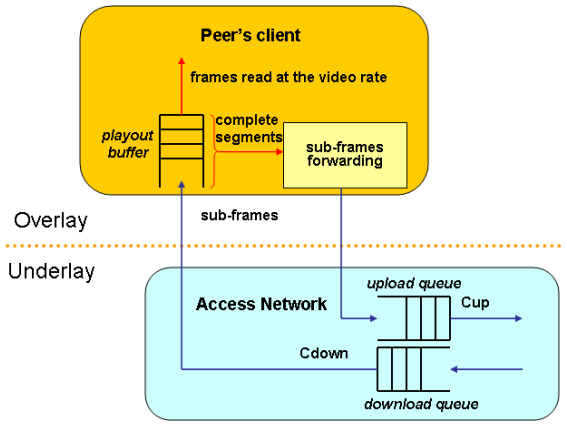

Fig. 2. Peer client and access network.

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. Standard 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 *standard join* procedure is described.

The *standard 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 *standard 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.

I. Optimized Join

This section describes a modification of the join procedure that can be put in place when *nearly-permanent peers* are available (see Section II.K).

When the *optimization* of the join procedure is enabled, new peers entering the system are placed at the highest position in each tree such that no peers with lower upload bandwidth are

at higher levels. The new peer replaces an already existing peer with smaller upload capacity. The existing peer is disconnected from the initial position and it starts a new join procedure. 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, in all the d trees for that offset, a parent is randomly chosen among those at the highest level in the tree having a sufficient amount of free upload bandwidth or at least one child with an upload capacity lower than the upload capacity $p_i.Cup$ of the current peer (a parent can be either another peer or the source).
4. If, for each tree, the chosen parent peer has not a sufficient amount of free upload bandwidth, its child with the lowest upload capacity is disconnected from the parent (if more than one peer have the same lowest bandwidth, the peer is randomly chosen among these).
5. After a time interval t_{join}, 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.

All the disconnected peers replaced by the new peer because of the rewarding optimization starts a new *optimized join* procedure for the tree they no more belong to. The nodes start receiving the sub-stream from their new parents after a time interval t_{rejoin_rew}. This interval models the time required for the identification and the selection of a new parent, when the rewarding procedure disconnects a peer.

J. Optimized Leave

The *optimized leave*, in addition to the activities prescribed by the *standard leave* procedure (see Section II.H), identifies an already existing peer for replacing the peer that has left the system. The leaving peer, at the depth level l in the tree, is replaced by considering all the peers connected to the same tree at the depth level $l+1$. Among all the peers at the depth level $l+1$ in the considered tree, the one with the maximum upload capacity is chosen (if more than one peer have the same highest bandwidth, the peer is randomly chosen among these), disconnected from its parent, and connected to the parent of the leaving peer. Then, the peer starts receiving the sub-stream from its new parent after a time interval t_{rejoin_rew}.

This optimization avoids the occupation, by new joining peers, of the free positions left by the leaving peers, when peers with higher upload capacity are already in the system. The jointly use of the *optimized join* and the *optimized leave*, in the following, is referred to as *rewarding*.

K. Nearly-permanent peers

In this section we investigate the effects of the availability of special nodes, referred to as *nearly-permanent peers*, on the overall quality of the peer-to-peer video streaming system. Nearly-permanent peers are peers whose *average permanence time* in the system, $1/\mu$, is more than one order of magnitude higher with respect to the duration of the media stream flowing into the peer-to-peer system or, from our point of view, with respect to the observation time of a simulation.

The reference system is accordingly completed with the following additional parameter:

1. *number of nearly-permanent peers*, N^{PERM}, that are permanently in the peer-to-peer video streaming system.

As a consequence, given the *average number of peers* N, the number of standard peers is $N - N^{PERM}$.

In our scenarios, two situations have been analyzed:

1. nearly-permanent peers are the N^{PERM} peers having the highest upload *capacity*;
2. the N^{PERM} nearly-permanent peers are peers *randomly* chosen among those in the system.

When the rewarding technique enabled by the *optimized join and leave* is used in the scenario with *nearly-permanent peers* having both a long permanence time and high upload bandwidth, the algorithm, when the system reaches its steady state, will place all nearly-permanent peers in the highest positions of the trees. This state is likely to be beneficial, since it allows the formation of short and stable trees.

III. Performance Analysis

In this section we study the impact of three critical system parameters on the performance of the peer-to-peer video streaming system, when nearly-permanent peers are used to optimize the system's performance.

The selected performance parameters are:

1. *average number of peers*, N, that are concurrently in the peer-to-peer video streaming system;
2. *number of nearly-permanent peers*, N^{SUP}, in the peer-to-peer video streaming system;
3. *average permanence time of peers* (excluding nearly-permanent peers) in the system, $1/\mu$, measured from peer first join to its leave.

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. When not otherwise explicitly notified, the used parameters are the following. 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 the *rewarding* technique is active, and consequently the *optimized join* and the *optimized leave* are used (see Section II.I and II.J), a *rewarding rejoin time* t_{rejoin_rew} equal to 500 ms has been considered. The *average permanence time of peers* has been considered equal to 15 minutes (excluding nearly-permanent peers). The average number N of simultaneously active peers has been considered 50, while the number of *nearly-permanent peers* N^{SUP} has been set to 10.

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 .

The capacity of the transmission/reception buffers in the access network is infinite, so that frame losses are not caused by buffer overflows.

A. Results

In this section, the performance indexes presented in the previous section are used to compare the behavior of the system, as the average number of peers, the number of available nearly-permanent peers, and average peer duration 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) Number of Peers

As shown in Fig. 3, the availability of nearly-permanent peers ($N^{PERM} = 10$) is only slightly beneficial if they are *random* peers. Conversely, if they correspond to the ones with higher *capacity*, the ratio of received sub-frames increases, compared to the scenario with no nearly-permanent peers, and depends on the considered average number of peers. For example, the gain is about 7% with 50 peers, and about 13% with 100 peers. The availability of nearly-permanent peers with high capacity allows mitigating the performance decrease because of the presence of a large number of peers, but is not sufficient to solve the scalability issue.

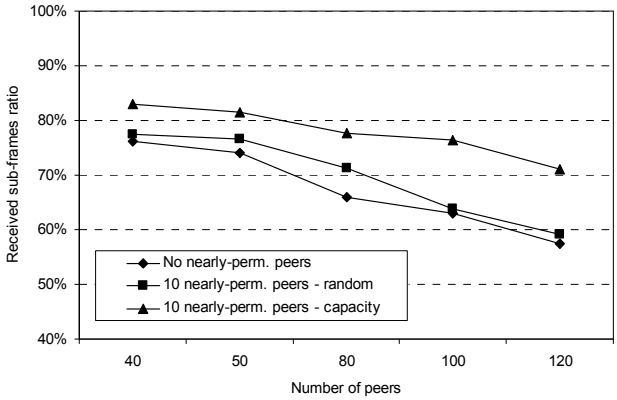

Fig. 3. Ratio of received sub-frames, with varying average number of peers, with zero and 10 nearly-permanent peers, chosen by capacity or randomly, without the use of rewarding techniques.

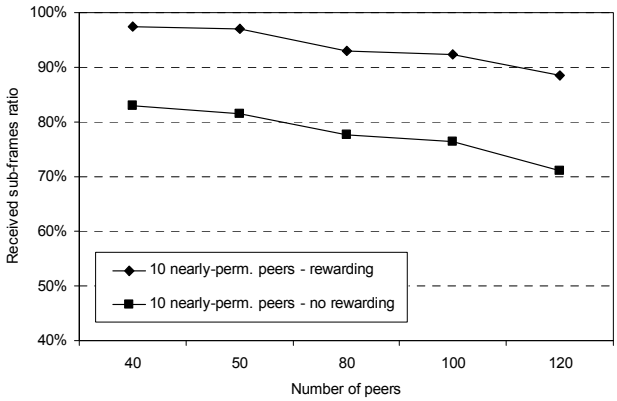

Fig. 4. Ratio of received sub-frames, with varying average number of peers and 10 nearly-permanent peers chosen by capacity, with and without the use of rewarding techniques.

If the rewarding technique is enabled (optimized join and optimized leave, see Sections II.I and II.J)), nearly-permanent peers – when the *capacity* criterion is used – allow gaining an additional 15% in the ratio of received sub-frames (see Fig. 4).

2) Average Permanence Time of Peers

As already investigated [14], smaller average peer durations cause lower performances. The availability of nearly-permanent peers helps improving the system's performance, but the amount of the improvement, as shown in Fig. 5, is not significantly affected by the average permanence time of peers.

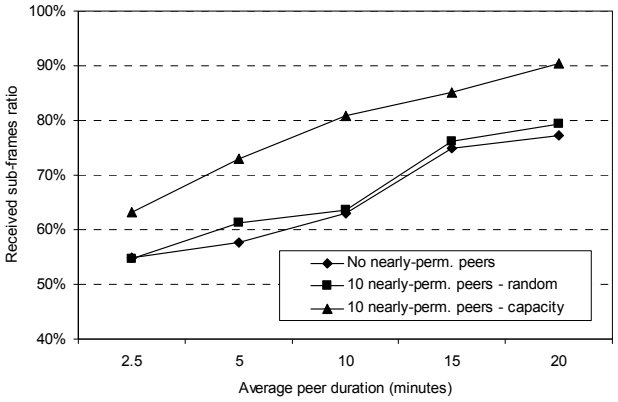

Fig. 5. Ratio of received sub-frames, with varying average permanence time of peers, with zero and 10 nearly-permanent peers, chosen by capacity or randomly.

3) Number of Nearly-Permanent Peers

Fig. 6 plots the quality of the received video stream as a function of the number of nearly-permanent peers, with and without the activation of the rewarding technique. Results indicate that, as the number of nearly-permanent peers grows, the ratio of received sub-frames increases.

In order to identify the optimal number of nearly-permanent peers for a given peer-to-peer video streaming system, we have tried a large number of combination of number of peers and number of nearly-permanent peers (i.e. $\left(N, N^{PERM}\right)$ pairs) able to provide a constant video quality. Fig. 7 shows the ratio of received sub-frames with varying average number of peers, when the rewarding technique is active. The ratio is around 94% for every point, and the amount of nearly-permanent peers is numerically reported below the corresponding points. Our finding is that a fixed percentage of nearly-permanent peers (in our scenario about 10-15%), with respect to the total number of peers, are required in order to assure a given quality of service.

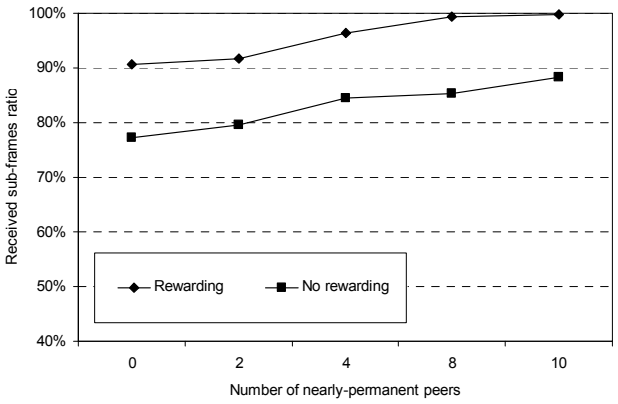

Fig. 6. Ratio of received sub-frames, with varying number of nearly-permanent peers chosen by capacity, with and without the use of rewarding techniques.

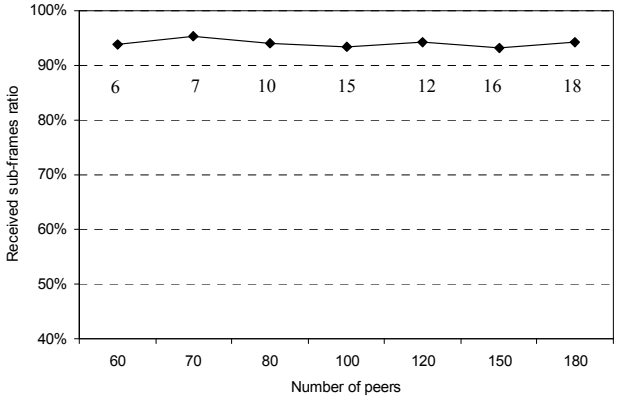

Fig. 7. Ratio of received sub-frames, with varying average number of peers, using the reported number of nearly-permanent peers chosen by capacity.

IV. CONCLUSIONS

In this work we then analyzed the effects of an efficient utilization of peers with high reliability and a long permanence time in peer-to-peer video-streaming systems with tree or forest topology. We have found that the presence of nearly-permanent peers allows increasing performances; however, these particular peers must be chosen according to their upload bandwidth, otherwise, their utilization is not significantly beneficial. We also discovered that the number of nearly-permanent peers needed to provide scalability is a percentage of the total number of concurrently active peers ranging in 10%-15%, in the examined scenarios.

Results obtained with this work can be beneficial to content providers in order to evaluate the cost/benefits tradeoff that peer-to-peer systems can offer. Moreover, results can be exploited by designers of peer-to-peer video streaming systems for increasing the efficiency and the overall performance.

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] Giacomazzi, P.; Poli, A. , "Performance Analysis of Peer-to-Peer Video Streaming Systems with Tree and Forest Topology," *Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on* , pp.287-294, Dec. 2008.