Performance analysis of mesh-based peer-to-peer video streaming systems

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 networks are an increasingly popular solution for the distribution of media content to a large number of users, with limited investments for network infrastructures. In this paper we focus on *mesh-based*, also referred to as *generic*, *unstructured*, or *data-driven*, peer-to-peer video streaming systems. We carry out a performance analysis of these systems, analyzing their sensitivity to the most critical system parameters. The analysis focuses on the number of neighbors of each peer, the average number of peers in the system, the permanence time of peers, and the amount of buffered video. Differently from existing studies we found that, depending on peers' permanence time, an optimal number of neighbors can be found. Moreover, moving towards a higher number of neighbors, the corresponding increase of signaling traffic causes a decrease of overall performance.**

Peer-to-peer; video streaming; mesh; generic; data-driven; performance analysis; neighbors

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, at no costs for content providers. 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] 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 unstructured overlay networks.

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

We analyze the conditions under which these systems can be reliable, and we provide a sensitivity analysis of critical parameters, such as the number of neighbors, the average number of peers, the permanence time of peers, and the amount of buffered video.

Most of existing studies focus on the analysis of full systems *as-is*, without investigating the impact on performances of changes in their operational parameters. Conversely, two interesting works in this direction are [13] and [14]. The former analytically analyzes the impact on efficiency – expressed in terms of bandwidth utilization – of a different number peers' neighbors. The latter provides analytical and simulative results about the percentage of received video depending on the buffer size and on the number of neighbors. However, [13] and [14] do not take into account the overhead due to signaling traffic, while [14] does not consider peers' dynamicity, i.e. the possibility for a peer to join and leave the system.

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 mesh-based peer-to-peer systems, each node periodically exchange information with other nodes, referred to as neighbors, about the availability of chunks of the video stream, in the following referred to as segments. Each node requests the segments it needs to one (or some) of its neighbors and reassembles the video stream. The topology of the overlay network is consequently represented by the logical mesh of peers interconnected by neighborhood relationships. Usually, in these systems the leave of a peer does not interrupt the data flow, since each peer has multiple potential providers of each segment.

Our model is inspired to CoolStreaming, a commercial implementation of the DONet protocol [4]. 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 not predetermined and data flow paths can vary dynamically, (3) at the application level, in the overlay peer-to-peer network, the content is organized as a sequence of chunks of video frames referred to as *segments*, (4) a single video *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, (6) each peer tries to keep a given minimum number of neighbors, (7) each peer tries to keep the number of neighbors below a given maximum, (8) as soon as a peer receives a complete segment it notifies all its neighbors, (9) every segment can be requested by a peer to a different neighbor.

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). 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}.

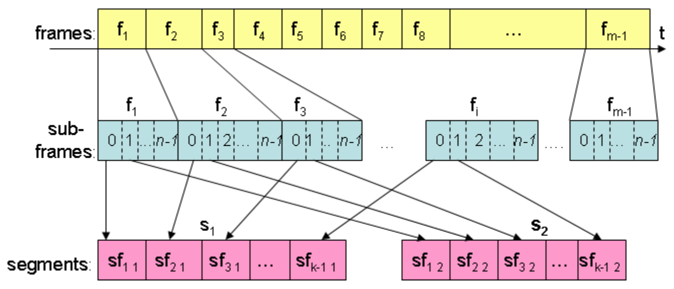

Fig. 1. Frames, sub-frames and segments.

C. BufferMap

Each peer keeps a set of n BufferMaps, one for each description[1] j. A BufferMap is a table where the identifiers of the available segments are stored. If the identifier of a segment is included in the BufferMap, the corresponding segment is available in the peer buffer (referred to as *playout-buffer* in the following), and can be requested by peer neighbors. BufferMaps are updated whenever a new segment is received or a segment is removed from the playout-buffer. Peers periodically send their BufferMaps, or subsets of the BufferMaps, to their neighbors. In our reference model, neighbors are informed about the arrival of a new segment as soon as it is received, i.e. the transmitted subset of the BufferMap delivers information about the last received segment.

D. MCache

Peers periodically receive information form their neighbors about the available segments. This information is stored by each peer in a set of n tables named Membership Caches (*MCaches*), one for each description j. *MCaches* store all the segments possessed by peer neighbors, and are kept up-to-date because of the periodic transmission of pieces of BufferMaps by neighbors. As a consequence, MCaches can be used for identifying all the neighbors having a particular segment.

E. Neighbors

Whenever a peer joins the system, it must connect to a set of neighbors in order to exchange the video stream. We model a system where a peer can get the list of available peers, either by a centralized or a distributed mechanism. A peer, for each description $j = 1, 2, \dots, n$, connects to a number of N_{MIN} peers, referred to as *neighbors*. A peer, whenever it detects the death of a neighbor, tries to identify a new peer in order to preserve the minimum number of N_{MIN} neighboring peers. A peer can accept requests from other peers until it reaches a maximum number of N_{MAX} peers. When the maximum is reached, all the requests related to the creation of neighborhood relationships are rejected, and requesters must contact other peers.

F. Source and Meshes

The video content is distributed among all peers through a set of q independent meshes. The number of meshes q is related to 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 through the same mesh.

The source provides the video stream by periodically creating new segments. In the same way as other peers, it accepts neighborhood requests until it reaches a maximum number of neighbors. Its BufferMaps are updated whenever a new segment is created or a segment is removed from the internal buffer; the source sends its BufferMaps, or subsets of the BufferMaps, to its neighbors. In our reference model, neighbors are informed about the presence of a new segment as soon as it is created, i.e. the transmitted subset of the BufferMap delivers information about the last created segment. The source has a limited amount of bandwidth Sup, measured in bit/s, and a maximum number of neighbors N_{MAX}.

G. Meshes and Peers

A peer, identified by the index p_i, in order to receive the video content, must be a node of the meshes carrying the content. A peer is not required to be part of all meshes. A peer can be placed in different positions in different meshes, and different meshes can have a different topology.

H. Peers Operations

Each peer is responsible for the distribution of the media. It operates both at the overlay (application) and the underlay (network) layer. The access bandwidth of peer p_i, expressed in bit/s, is referred to as $p_i.Cup$ and $p_i.Cdown$; these parameters represent the upload and the download capacity of its access network link, respectively. The peer is provided with an underlay transmission buffer (the *upload queue*) and an underlay reception buffer (the *download queue*).

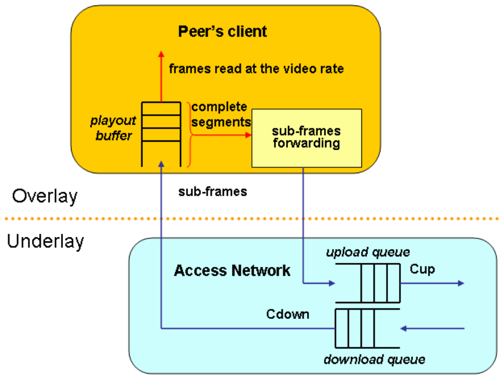

Fig. 2. Peer client and access network.

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

1. All sub-frames received from the download queue are stored in the overlay playout buffer, where the stream is reassembled, according to frame numbers and sub-frame offsets. As soon as all the sub-frames forming a segment are received, the BufferMap is updated with the availability information, and neighbors are informed with the transmission of a subset of peer BufferMap.
2. All *BufferMaps*, or subsets of BufferMaps, received from peer neighbors, are used to update *MCaches*. If the updated information in a MCache is related to segments not already requested by the peer, a *countdown timer* is started. During the countdown, BufferMaps from other neighbors, related to the same segment, can be received. When the timer expires, a neighbor is chosen among all the neighbors having the

[1] A description is a sub-stream of a single media stream that is coded with a Multiple description coding (MDC). The segments of each description are routed over multiple independent overlay meshes.

segment. A *request packet* is then inserted in the upload queue of the peer and sent to the selected neighbor.

3. All the received *request packets* are processed, and the requested segments are copied in the upload queue and sent to the requesters.
4. The frames stored in the playout buffer are sequentially extracted by the client's player at the rate of the video stream.

I. 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, the related BufferMap is updated and the availability information is sent to the neighbors of the specific description (sub-frame offset). When all the frames of a segment have been read by the player, a position in the playout buffer is freed and the related BufferMap is updated.

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.

J. Join

When a new peer wants to receive the video stream, it must become part of one or multiple distribution meshes. In the following, the *standard join* procedure is described.

The *join* procedure connects new peers to a number of N_{MIN} peers, referred to as *neighbors*. 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 randomly selects a number of N_{MIN} peers, referred to as *neighbors*. The selected neighbors accept the request if their current number of neighbors is not above the maximum threshold N_{MAX}. If the maximum threshold is reached, the request for the creation of the neighborhood relationship is rejected, and the peer must contact another randomly chosen peer.
4. After a time interval t_{join}, measured in seconds, the new node starts receiving the BufferMaps from its neighbors. This interval models the time required for the identification and the selection of the neighbors.

K. Leave

A peer can leave the system without notification. Whenever this happens, its neighbors cease to receive the information about the available segments (BufferMaps), and they try to identify new neighbors if their current number of neighbors is below the minimum threshold N_{MIN}. The neighbors identified for the replacement of dead peers, accept the request if their current number of neighbors is not above the maximum threshold N_{MAX}. The peers start receiving the *BufferMaps* from their new neighbors after a time interval t_{rejoin}. 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 neighbor and the subsequent search for a new candidate neighbor.

III. ANALYSIS OF CRITICAL PARAMETERS

In this section, we analyze a peer-to-peer video streaming system where peers dynamically join and leave. The system, in the steady state, has an average number N of simultaneously active peers. The time spent by a client in the system is exponentially distributed with average duration equal to $1/\mu$ s. Joins are independent Poisson events with a total average rate of joins equal to $\Lambda\ s^{-1}$, such that $N = \Lambda/\mu$.

A. Parameters and Indexes

In this section we study the impact of five critical system parameters on the performance of the peer-to-peer video streaming system described in Section II. The selected performance parameters are:

1. *number of neighbors range*, $N_{MIN} - N_{MAX}$, representing the minimum and maximum number of neighbors of a peer;
2. *average number of peers*, N, present in the peer-to-peer video streaming system;
3. *provider choice criterion*, used to choose among the neighbors having the required segment of video (i.e., how a peer selects the provider of a segment in a set of multiple neighbors);
4. *average permanence time of peers* in the system, $\frac{1}{\mu}$, measured from the peer's first join to its leave;
5. *playback threshold*, $PBTh$, that must be reached in peer playout buffer before the video stream is locally played.

In this work we have concentrated our attention on system features with a significant impact on performance, not covered in depth by extant literature, and critical for the evaluation of the efficiency of mesh based peer-to-peer video streaming systems. Notice that other system parameters might be as well critical; the study of additional parameters is left for further research.

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.

Additional information is also registered, such as the number of total joins and leaves, and the topological characteristics of the overlay distribution network.

B. Simulation Parameters

The system is analyzed through a simulation tool expressly implemented for this purpose. We have carried out an extensive simulation process by considering a peer-to-peer video streaming system with mesh topology fed with a video trace 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 independent mesh. 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 depth has been set equal to 133.3 s ($PBlength = 1800$), and we have used a playback threshold $PBTh$ of 20 s. The join time t_{join} has been set to 500 ms, and the rejoin time t_{rejoin} equal to 100 s. When not otherwise explicitly notified, the average number N of simultaneously active peers has been considered 100, the *average permanence time of peers* has been set to 15 minutes, and the *provider choice criterion* has been by *free capacity*, i.e., a peer request segments to the peer with the largest amount of free upload bandwidth among all its neighbors.

The upload and download access bandwidth ($p_i.Cup$ and $p_i.Cdown$) for joining peers have been set to $Cdown = 7$ Mbps and $Cup = 2$ Mbps .

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

C. Results

In this section, the performance indexes presented in Section A are used to compare the behavior of the system, as the number of neighbors range, average number of peers, rejoin time, average permanence time of peers, and playback threshold 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 Neighbors Range N_{MIN} - N_{MAX}

The number of neighbors in peer-to-peer systems with mesh topology is critical, since few neighbors could not guarantee the availability of a provider for all the required segments of the video stream, especially when one or more neighbors leave the system. Conversely, a high number of neighbors could introduce a high transmission overhead because of the continuous exchange of BufferMaps among the peers.

The bidimensional set of possible $\{N_{MIN}, N_{MAX}\}$ values has been explored by means of a large number of simulations, and the optimal pair of values has been determined to be {4, 9}. We discovered that these optimal values do not depend on the capacity of the access network, and are also optimal for different values of the average number of peers N .

In the following figures, the results obtained by using a homogenous distribution of peers access bandwidth equal to $Cdown = 7$ Mbps and $Cup = 2$ Mbps , and the average number of 50 peers, are shown.

TABLE I. OVERALL OVERHEAD WITH VARYING N_{MIN} AND N_{MAX}.

N_{MIN}	N_{MAX}	Overhead
2	9	4.6%
3	9	5.5%
4	9	5.9%
5	9	6.4%
8	9	7.1%

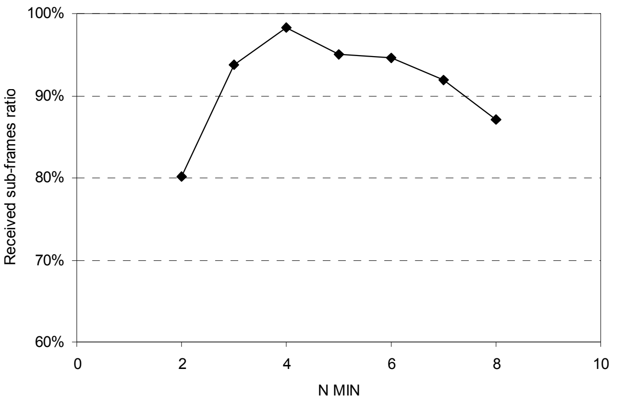

Fig. 3. Ratio of received sub-frames with N_{MAX}=9 and varying N_{MIN}.

Fig. 3 plots the percentage of received sub-frames as a function of the minimum number of neighbors N_{MIN} a peer is required to have, when N_{MAX} is kept fixed to 9. A minimum value of neighbors below 3 causes a degradation of performance, since the potential providers of segments are few, multiple paths for the video stream cannot be exploited and the system is sensitive to the departure of peers. Conversely, if the number of neighbors grows, the overhead caused by the exchange of information among peers has a negative impact on performance. We also found that the actual average number of neighbors per peer is approximately equal to $N_{MIN} + 1$. Table I reports the overall overhead determined by *BufferMap* messages and by *request packets*, with different values of N_{MIN}.

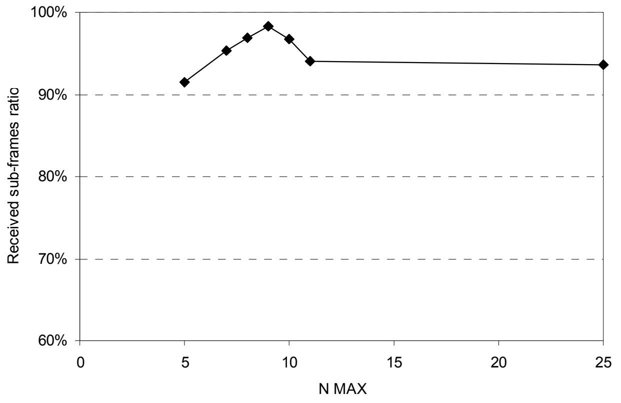

Fig. 4. Ratio of received sub-frames with N_{MIN}=4 and varying N_{MAX}.

If the maximum number of neighbors that a peer can accept is below 9, as shown in Fig. 4, performances decrease because peers refuse too many connections from other peers. Values higher than 9 do not significantly reduce the quality of the reception of the video stream, since the actual number of neighbors is influenced essentially by N_{MIN}.

2) *Average Number of Peers and Provider Choice Criterion*

We explored two different criteria for the choice of the neighbor the peers send the request to:

1. *random criterion*, i.e. the neighbor is chosen randomly;
2. *free capacity criterion*, (see Section B). The free upload bandwidth is evaluated as an average on the previous 5 s interval.

Fig. 5 plots the ratio of received sub-frames as a function of the average number of peers in the system, when the two different *provider choice criteria* are used. The use of the *free capacity criterion* assure higher performances, although only about 1% higher; the use of the simpler *random criterion* can consequently be implemented with not significant disadvantages if compared with a more complex algorithm.

As far as the number of peer is concerned, the figure shows that an increase in the number of peers also negatively impact on overall performances. However, moving from 20 peers to 200 peers has only a negative impact equal to 3% of received sub-frames.

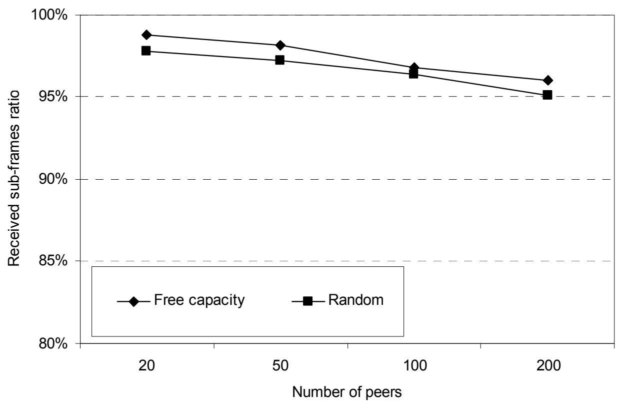

Fig. 5. Ratio of received sub-frames with varying average number of peers, using different provider choice criteria.

3) *Average Permanence Time of Peers*

Fig. 6 shows that when the life of peers is short the quality of the video stream is lower. This is a consequence of the high number of neighbors that die, causing temporary decreases of the number of potential suppliers of a segment. We observed that performances decrease when the average peer duration is above 10 minutes. This is because with a longer duration of peers, the actual average number of neighbors per peer is larger and the signaling overhead increases.

The degradation of performances due to a large rate of peer leaves can be mitigated by using higher values of N_{MIN}, so that more potential segment suppliers can replace dead peers. The degradation of performances at high peer durations can instead be solved by limiting the number of neighbors by means of the N_{MAX} parameter. We tried to identify the optimal values of N_{MIN} and N_{MAX} in the different scenarios; results are shown in Fig. 7.

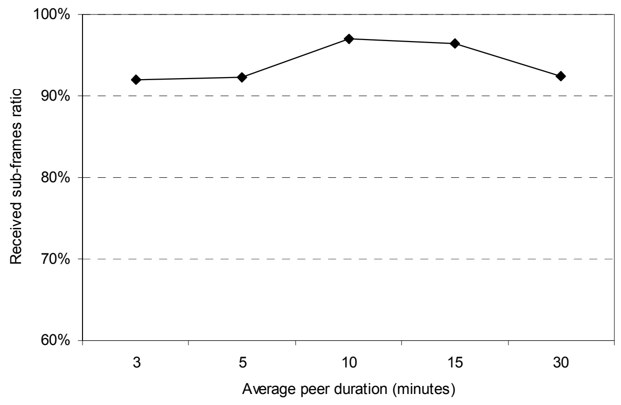

Fig. 6. Ratio of received sub-frames with varying average permanence time of peers, using N_{MIN}=4 and N_{MAX}=9.

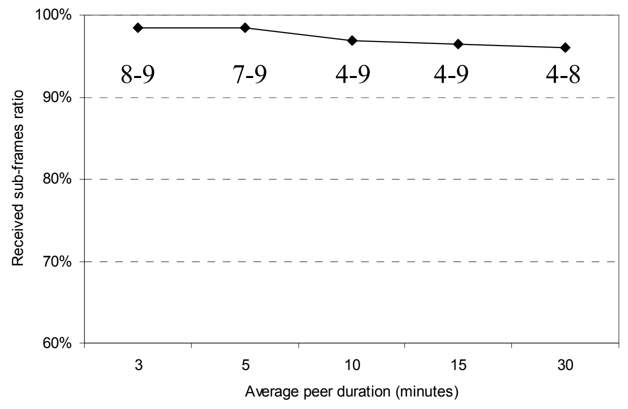

Fig. 7. Ratio of received sub-frames with varying average permanence time of peers, using the optimal N_{MIN}-N_{MAX} pairs (reported below the points of the curve).

The average peer duration of 3 minutes required a minimum number of peers equal to 8, two times the optimal value of the 10 minutes duration case. Conversely, peer duration of 30 minutes required to decrease the maximum number of neighbors from 9 to 8, in order to keep the overhead in an acceptable range. As a consequence, we found that the $\{N_{MIN}, N_{MAX}\}$ pair should accurately be configured according to the expected stability of the peer-to-peer system.

4) Playback Threshold

The playback threshold represents the amount of buffered video that must be reached before the video stream is locally played. Results of several simulations show that performances are maximized when the amount of buffered video is more than a given threshold, in our scenarios equal to 20 s. The mesh topology of the peer-to-peer video streaming system does not prearrange the paths followed by the segments from the source to peers; as a consequence, a large playback threshold is required in order to allow the arrival of segments by means of multiple meshes and multiple paths, with different delays.

IV. Conclusions

The number of neighbors in peer-to-peer system with mesh topology is critical, since few neighbors can not guarantee the availability of a provider for all the required segments of the video stream, especially when one or more neighbors leave the system. Conversely, a large number of neighbors could introduce a high transmission overhead because of the continuous exchange of information among peers.

These results significantly contrast with [13] and [14], which prove that higher numbers of neighbors can assure better performances. This conclusion is obtained since the signaling traffic overhead (BufferMaps exchange, and request packets transmission) is not taken into account in [13] and [14].

We discovered that the minimum and maximum number of neighbors should be customized according to the expected stability of the peer-to-peer system, using larger number of neighbors when the system is less stable.

Our results also show that the choice of the provider of a segment among all the neighbors by their free capacity criterion is not significantly beneficial with respect to the use of the simpler random criterion. We discovered that the amount of video to be bufferized by every peer should be above a threshold, equal to 20 s in our simulations, significantly higher compared to systems with tree topology, when a 5 s threshold is typically enough [15].

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] Hao Liu; Riley, G., “How Efficient Peer-to-Peer Video Streaming Could Be?,” *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pp.1-5, Jan. 2009.
[14] Ren, Zhenfeng; Liu, Ju; Qin, Fenglin; Wang, Yanwei, “A Survey on Peer-to-Peer Streaming System's Neighbor Number,” *Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on*, pp.421-424, Oct. 2009.
[15] 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.