Comprehensive Study of Buffering Mechanisms in Hybrid Live P2P Streaming Protocol HLPSP

Chourouk Hammami*, Achraf Gazdar† and Abdelfettah Belghith†
HANA Lab., University of Manouba, Manouba 2010, Tunis, Tunisia
Email:chourouk_ham@yahoo.fr
†College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
Email: agazdar@ksu.edu.sa, abelghith@ksu.edu.sa

***Abstract*—Buffering mechanisms play a vital role in leveraging the capacity and efficiency of live streaming as well as P2P streaming system. An appropriate buffering mechanism allows the peer to undergo a smooth playback by compensating the delay introduced by the network as well as storing pieces to be played in the near future. A circular buffer is used most of the time in a live P2P streaming system as the media duration is not known in advance. When the speed of the downloading process exceeds the speed of the playback process, some not-yet-played pieces my be erased which affects in turn the playback quality. In this paper we investigate this problem through extensive simulations done without loss of generality in the context of our recently proposed P2P protocol named HLPSP and we propose a pieces requesting control mechanism to avoid such a problem. The simulations results show that the proposed mechanism eliminates the erasing phenomena which decreases the rate of the missed pieces during the playback process.**

I. Introduction

With the widespread penetration of broadband accesses, multimedia services are getting increasingly popular among users and are contributing to a significant amount of todays Internet traffic [1], [2]. Many popular commercial video streaming applications such as YouTube are based on content delivery networks (CDNs). Content is pushed to a set of strategically placed content delivery servers and consumers then stream content from nearby servers. A CDN-based solution avoids the bottleneck of a central server, can achieve lower streaming latency, reduces network traffic and can serve more users. However, it usually inflicts high deployment and maintenance costs. The current estimation of YouTube's costs is 1 million dollars per day [3] and those costs could increase drastically if more videos continue to be switched to higher qualities. Peer-to-Peer (P2P) networks enable efficient resource sharing in distributed environments and provide a highly effective and scalable solution to this problem. Such approach eliminates dedicated servers to mediate between end systems and exploits client resources to forward the media data and to offer a flexible and scalable solution for streaming applications. Actually, various P2P streaming systems have been proposed to enable live and video-on-demand (VoD) streaming to large audiences with better video quality at low deployment and server costs [4]. Throughout the paper, we will focus on the buffering mechanism used in live streaming systems based on P2P overlay networks.

There are three widely adopted overlay architectures in P2P-based live streaming systems: the tree mechanism, the mesh mechanism and the hybrid mechanism. All these architectures aim to perform desired performance properties, such as having overlay paths with low latency and high bandwidth. Indeed, more popular approaches to P2P Streaming is using mesh topology among peers. The key aspect of the design where mesh-based differs from tree-based approaches is the lack of a formal structure for delivering data. More explicitly, a video stream is divided into pieces of a uniform length, and the availability of the active pieces in the buffer of a node is represented by a Buffer Map (BM). Each node continuously exchange its BM with its partners, and then determines which pieces is to be fetched from which partner accordingly. Several systems exist that use this approach to content streaming, and notable examples are CoolStreaming [5]that was one of the first systems to employ this approach, and some that are highly popular in East Asia such as PPLive [6], QQLive [7] and TVAnts [8].

The main goal of this paper is to investigate the buffering parameters of HLPSP [9], [10] that implements the same buffering mechanism as Coolstreaming. We readily observe through extensive simulations that the buffer size (BS), the frequency of exchange of Buffer (BPE) and the number of buffered pieces prior to the playback (NBP) affect considerably the performance of the protocol. We also detected that the pieces missed by the playback process is due to the network download lateness as well as the speed of the downloading process which may erase the not-yet-played pieces in the buffer whenever it exceeds the speed of playback process. Towards this issue, we take these three parameters into consideration in this paper to optimize the buffering mechanism in such way that we eliminate the erasing phenomena during the download of the future pieces (to be played in the future).

The rest of this paper is organized as follows: In section 2, we give the related works in this area. Section 3 presents an overview of the buffering mechanism in HLPSP. Major experimental results and analysis are presented in section 4. Our proposed pieces requests control mechanism in HLPSP is given and evaluated in section 5. Section 6 concludes the paper and briefly discusses our future research plan.

II. Related Works

In this section, we describe the related works in the context of the P2P live streaming systems as well as the buffering mechanism used in such systems.

A. Live streaming approaches based P2P

The existing P2P overlay multicast approaches for live streaming can be classified roughly into two categories: tree-based overlay multicast and mesh-based P2P overlays. In the tree-based approach, the key is to construct a multicast tree among end hosts. The End System Multicast examined the main difficulties encountered in the native IP. The construction of a multicast tree was done using an overlay network. So, a tree protocol organizes peers into a tree rooted at the source server. In this structure, nodes without children are called leave nodes, which are neither the root node nor the leaf nodes, are called branch nodes. Streaming flows originating from the tree root will go down along the branch nodes, and finally reach the leaf nodes. NICE [11] and ZigZag [12] are examples of this kind of protocols. This single-tree based overlay suffers from two drawbacks:1) potentially poor resource usage and unfair contributions since a leaf node cannot contribute with its upload capacity; 2) given the dynamic nature of nodes, the departure or failure of high-level node can cause significant program disruption and requires the re-construction of the overlay topology. The multi-tree approach [13], [14] was introduced to tackle single-trees problems where the source encodes the stream into sub-streams and distributes each substream along a particular overlay tree. There are two key improvements done by the multi-tree solution. First, the overall system is more resilient, as a node is not completely affected by the failure of an ancestor on a given tree. Second, the bandwidth of all nodes can be more fairly utilized, as long as each node can only be a leaf of one tree. However, a multi-tree scheme is more complex to manage in the presence of network dynamics. To improve the stability of services, mesh-based protocols have been proposed; each peer can accept media data from multiple parents as well as providing services for multiple children. Meshes based on Gossip protocol can find fresh peers in the single mesh. So, this structure is highly resilient to node failures, but it is subject to unpredictable latencies due to the frequent exchange of notifications and requests. PPlive [6], DONet/Coolstreaming [15], [16] and Prime [17] are examples of mesh based systems. To improve performance, some hybrid systems combine tree and mesh structures to construct a data delivery overlay such as [9], [10], [18], [19], [20], [21], [22], [23], [5]. For instance, the key idea of HLPSP [9] is to organize the nodes on levels based on their uploads' capacities where the data flow: 1) from the higher to the less level following a tree-based topology and 2) following a mesh topology within the same level. Hence, unlike the other hybrid solutions, within HLPSP, we can find more than only one especial node within the same level which increases the chance to continue solutions, within HLPSP, we can find more than only one especial node within the same level which increases the chance to continue downloading data from the other nodes even if some nodes leave the network and moves the powerful peers close to the source which allows minimizing the end-to-end delay (maximize the freshness) and serving the maximum of peers in a stable environment. It organizes the nodes on levels based on their upload capacities. The source of the media belongs to the highest level 0. Depending on their upload capacities, the nodes are classified within several levels varying from level 1 (the higher level) to level x (a lower level) where $x > 1$. Each node can be served by nodes belonging to the same or to a higher level called active nodes. A node in a given level can serve nodes of the same or less level called passive nodes.

Three actors are involved in HLPSP: the media source, the tracker and the peers. Initially, the source sends its upload capacity to the tracker for registration. Then, the later calculates the maximum number of its passive neighbors. Hence, the tracker can receive the demands sent by the peers (Neighbour request). These requests are sent in two cases: a) during the peer connection; b) the number of current active neighbors of a peer is less than its effective neighbours, an effective neighbor is an active neighbor who really supplies the concerned peer by the pieces. In the reception of a Neighbour Request from a peer, the tracker responds by a Neighbour Response which contains the list of its available active neighbors. Once the peer receives the tracker response it starts establishing its neighborhood relations. Three types of messages can be exchanged between peers: join request, join response and join deny.

Periodically (1 second), each peer sends to its passive neighbors a buffer map message containing the list of the media pieces available in its buffer. Based on the segments availability information periodically exchanged between the node and its partners, each node is able to schedule which segment is to be fetched and from which partner accordingly. Two messages are exchanged between peers to download the media pieces: 1) piece request sent by a peer to an active neighbor to ask for a data piece; 2) piece response which contains the requested piece as a response to the previous request message. For more details about HLPSP, the reader can refer to [9].

B. Buffering mechanisms

Buffer control is indispensable to a P2P network, and it is also one component of P2P live streaming system to avoid jitter, especially in swarming delivery under randomly mesh architecture. Two reasons could be stated on the importance of using buffers in mesh structure. First, the buffer keeps video playing smoothly. Second, the buffer mitigates the possible challenges because of P2P churns. In general, peer churn often causes an interruption or a break of data delivery; hence, the buffered data can deal with the emergency before the overlay recovery.

There are lots of commercial P2P streaming systems on the internet, such as PPLive [6], PPStream [24] and UUSee [25]. Despite the remarkable popularity, it is still a key issue for both researchers and engineers to optimize the buffering mechanism. As examples, the buffer size is set to 120s in PPLive [6] while only 21s is used in PPStream [24]. M. Zhang et al. [26] uses 60s as buffer size. In [27], it is found that buffering can dramatically improve performance of P2P streaming system and a 30-second lag is sufficient to obtain almost all potential gains from buffering. According to [28], buffer map exchange period accounts for most of the performance gap that separates the actual and optimal performance of pull-based mesh protocols. Above works study the impact of either BS or BMEP on system performance and thus the relationship between the BS and BMEP is not fully revealed. In addition, these works did not considered the third key parameter (NBP) of the buffer mechanism. Towards this end, we take BS, BPE and NBP together into consideration in this paper to optimize the buffering mechanism.

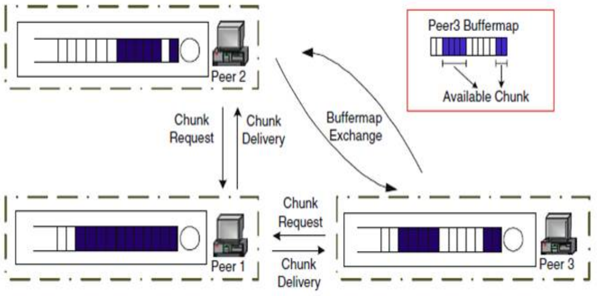


Fig. 1: Buffer map exchange and data pull among peers

III. Overview of the buffering mechanism in HLPSP

HLPSP has the ability to effectively utilize the uploading bandwidth of all participating peers as the group size grows. The data unit for the streaming is the media piece. Each piece contains media content for a small time interval. Each piece has unique timestamp and sequence number. Since pieces may arrive at receivers out of order, these values are used to re-order pieces for playback. HLPSP couples push content reporting with pull content requesting. Each node has a storage buffer for the latest received trunks and a buffer map of available trunks in the buffer. The pushing content reporting is performed when a node sends its buffer map to its neighbors upon receiving new pieces while the pulling content requesting takes place when the node asks for new pieces.

Figure 1 illustrates a scenario where buffer maps are exchanged and needed pieces are pulled between peers. Given the local BM of a node and the remote BMs of its partners, a schedule is then generated for fetching the expected pieces from the partners. Specifically, the scheduling algorithm strikes to meet two constraints: the playback deadline for each piece, and the heterogeneous streaming bandwidth from the partners. If the first constraint cannot be satisfied then the number of pieces missing should be kept as minimum as possible to maintain a continuous playback.

The buffering mechanism helps to provide smooth playback in streaming P2P networks. But the HLPSP has to pay a price for its buffer map advertising. Towards this issue, we focus on optimizing the buffering mechanism.

The core of buffering mechanism in P2P live streaming system is BS, BPE and NBP. Intuitively, increasing buffer means that peers have to devote more memory space; however, decreasing buffer means that the system performance may be degraded. Besides, shortening BPE must improve the system performance but at the same time causes much addition signaling overhead, which means that an important part of the upload bandwidth of the P2P network is wasted due to the high exchange of the buffer map messages. In addition, the peer needs some NBP before starting the playback of the stream in order to ensure smooth playback without interruptions caused by the delay variations (Jitter) in the network. In fact, large NBP degrades the playback delay but smaller NBP deteriorates the continuity. But as we will show later in the paper, BS, BPE and NBP are not the only parameters that can affect the P2P performances but also an uncontrolled pieces request process can deteriorate the playback experience of the user when it is done with a speed exceeding the playback rate. All these matters will be presented and discussed in the next sections.

IV. HLPSP Performances' Evaluation

In order to study the impact of the buffering mechanism on the performance in HLPSP, we carried out a set of simulations, in different scenarios using the discrete event network simulator, OMNET++ simulator [29]. The Network infrastructure used in the simulations is generated by GT-ITM module [30]. We are interested on the investigation of the key parameters in the buffering as BS, NBP and BPE. To this end, we varied the BS, BPE and NBP which are set in HLPSP to: 1000 pieces, 1s and 300 pieces respectively. We considered StarWars IV trace file to simulate the real behavior of the stream. The trace file is obtained from the Video Trace Library. The streaming rate is set to 512 Kbps. The video stream is decomposed into several pieces with an average size almost equal to 130 KB. The Nodes start playing the stream after 40 seconds buffering time. This is comparable to the most widely deployed P2P live streaming system such as Sop-Casts where the average startup time is about 30-45 seconds [31]. For all experiments, only one media source is considered with an upload bandwidth equal to 5 Mbps. We try to determine the performance of the protocols in more realistic settings. In fact, most peers are home users, so we focus on the ADSL technology, currently a prevalent deployment technology for Internet access from homes. Since target p2p live streaming users are individual persons using mainly lowest offers for the ADSL technology, while the highest bandwidth offers are most used in the professional field by enterprises, we adopt different peer capacities as following: 10% (down 24 Mbps, up 2 Mbps), 10% (down 20 Mbps, up 1,5 Mbps) 10% (down 12 Mbps, up 1,3 Mbps), 15% (down 8 Mbps, up 1,2 Mbps), 25% (down 4 Mbps, up 1,1 Mbps) and 35% (down 2 Mbps, up 1 Mbps). Table I summarizes our simulation parameters.

TABLE I: Simulation parameter's list

Parameter	**Value**
Maximum packet size	1000 Bytes
Buffer size	Variable, range from 500 to 2000
Buffermap exchange period	Variable, range from 0.5s to 1s
Video codec	MPEG4 Part I
Video FPS	25
Number of frames in GoP	12 frames
Selected trace file	Star Wars IV
Average size of piece	130 Kb
Average video bit rate	512 Kbps
Simulation duration	500 s
Capacity of peers	10% (down 24 Mbps, up 2 Mbps) 10% (down 20 Mbps, up 1,5 Mbps) 10% (down 12 Mbps, up 1,3 Mbps) 15% (down 8 Mbps, up 1,2 Mbps) 25% (down 4 Mbps, up 1,1 Mbps) 35% (down 2 Mbps, up 1 Mbps)
Number of passive neighbours	Min (threshold(10), upload/piece size)
Number of active neighbors	Min (threshold(10), download/piece size)
Number of levels	6
Number of runs	10
Source number	1

A. Simulation set up

During the simulation, we focus on the following criterions:

missed pieces: The percentage of video content that is lost over the original video;

Playback delay: The average delay between the playback time in the peers side and the streaming time on the server side;

Erased pieces: The number of pieces erased before the playback.

We adopted the following simulation scenario. Initially, 700 nodes join the network during the period [0s,20s], according to a uniform distribution with an inter arrivals μ between [15ms,20ms]. After, 490 nodes accidently leave the network during the simulation period [30s, $Simulation\ Duration - 20s$] uniformly with the same parameter μ. Simultaneously, 300 nodes uniformly join the network with the same inter arrivals μ.

B. Simulation results

From figures 2a, 2b and 2c , it is observed that increasing BS can continuously increase the continuity. For example, for a fixed NBP and BPE (300, 1s) respectively the data loss rates is 14% when BS is 1000 and only 5% when BS is 2000. In fact, if BS is too small, the piece in peers buffer is removed too early and thus some peers can not retrieve certain pieces

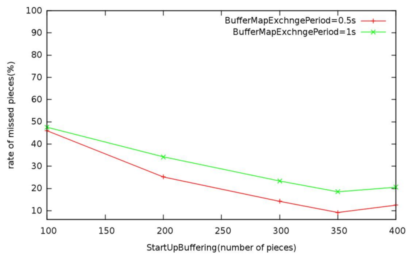
However, figures3a,3b and 3c show that the playback delay can be slightly affected by the increase of BS.

For all the proposed buffers sizes, we readily observe that shortening BPE can continuously decrease the data loss rate and the playback delay provided by HLPSP. For example, we varied BPE for a fixed NBP (300) and fixed BS (1000). Thus, we obtained the following results: when BPE is set to 0.5s the missed pieces rate is 11% and the playback delay is 24s. But when BPE is set to 1s the missed pieces rate is 22% and the playback delay is 27s. It can be concluded that BPE is the key parameter to shorten peers playback delay and the missed pieces rates. Small BPE means that pieces can be distributed among peers more quickly and thus the playback delay is shortened and the fluidity is improved.

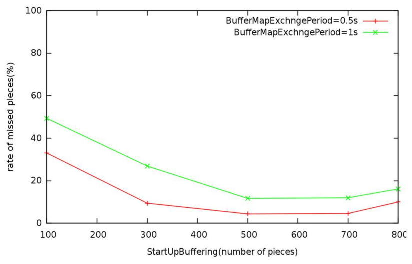
From figures 2a, 2b and 2c, we can see that increasing NBP can improve the system performance. However, if NBP reaches a certain threshold, increasing NBP induces a negative impact on the system performance. When NBP is less than 70% of the buffer size, the increase of NBP can greatly shorten the missed pieces rates and the playback delay. However, increasing NBP has a negative influence on fluidity when the NBP is larger than 70% of the buffer size. In this case,the download speed exceeds the playback speed. So, some pieces will be erased before their reading as a circular buffer is used most of the time to save future pieces (not yet played). The figures 4a, 4b and 4c confirm our ascertainment. Indeed, for all the buffer sizes the number of erased pieces begins to appear when the NBP is 50% of the buffer size and becomes important when the NBP is equal to 70% of the buffer size.

V. Our proposed algorithm

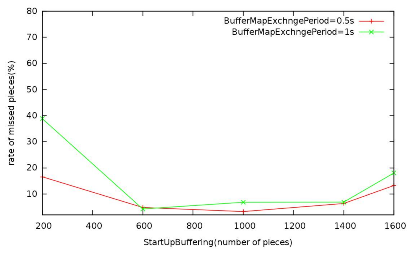
Our proposed solution aims to slow down the download speed with regards to the playback speed while trying to



(a) Buffer size = 500



(b) Buffer size = 1000

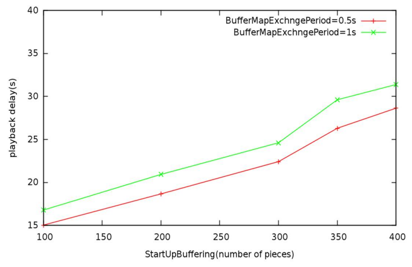


(c) Buffer size = 2000

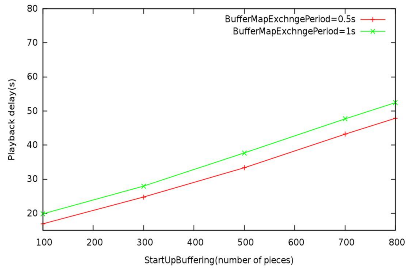
Fig. 2: Rate of the missed pieces as a function of the start up buffering for HLPSP

download the maximum future pieces (to be played in the future). As portrayed by figure 5, the maximum requested pieces ($MaxNumPieces$) shouldn't exceed the empty part of the buffer ($PiecesNotDownB$) plus the already played part ($PiecesPlayB$). Algorithm 1 details how this requests control mechanism has been implemented in HLPSP. As detailed in the algorithm, the peer starts by discovering the new available pieces in his neighbours bitmaps ($RemoteAvailablePieces$) then it checks his buffer to know the number of available entries. Based on this check routine, the peer calculate $MaxNumPieces$ of pieces to be requested which is equal to the minimum value between the remote available new pieces ($RemoteAvailablePieces$) and the available entries in his buffer ($PiecesNotDownB + PiecesPlayB$). Recall that we suppose that a circular buffer is used to download the future pieces which may lead to erasing some not-yet-played pieces if the download process progresses faster than the playback process.

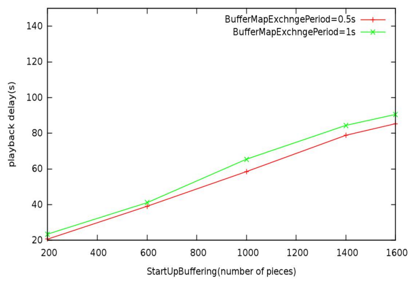
To evaluate the impact of the proposed request control mechanism on the HLPSP performances, we carried out the same simulations scenarios presented in section IV. The simulation results are plotted in figure 6 where HLPSPo



(a) Buffer size = 500



(b) Buffer size = 1000



(c) Buffer size = 2000

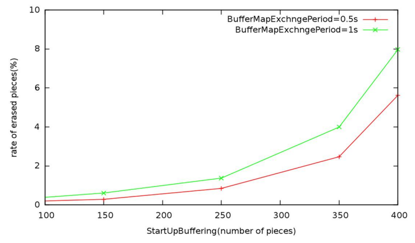
Fig. 3: Playback delay as a function of the start up buffering for HLPSP

Algorithm 1: Calculating the maximum number of pieces to request

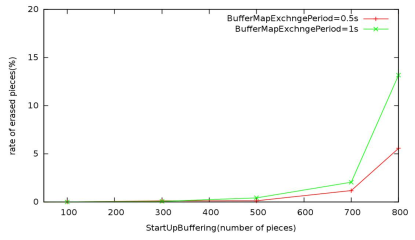
```
Require: Timer to send pieces requests elapses
  $RemoteAvailablePieces \leftarrow$
  $getAvailablePiecesFromRemoteBitmap()$
  $MaxNumPieces \leftarrow$
  $min\{RemoteAvailablePieces, PiecesNotDownB$
  $+PiecesPlayB\}$
  $i \leftarrow 0$
  while $i < MaxNumPieces$ do
      $SendNextPieceRequest()$
      $i \leftarrow i + 1$
  end while
```

designate HLPSP with the pieces request control mechanism previously described in algorithm 1.

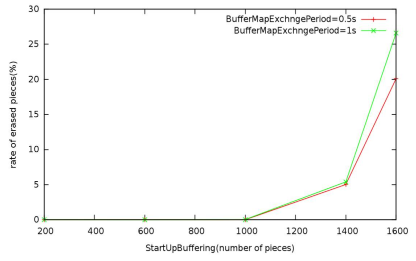
As we can see in figure 6a, our mechanism decreases the rate of missed pieces by reducing the erased pieces number to 0 as shown by figure 6b. Only the piece download lateness caused by the network is counted in the total missed pieces (not downloaded at time during the playback time).



(a) Buffer size = 500

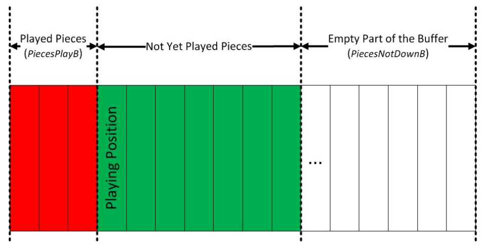


(b) Buffer size = 1000



(c) Buffer size = 2000

Fig. 4: Rate of the erased pieces as a function of the start up buffering for HLPSP

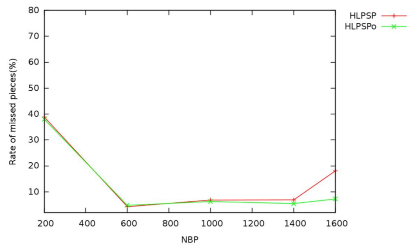


Played Pieces (PiecesPlayB)
Not Yet Played Pieces
Empty Part of the Buffer (PiecesNotDownB)
Playing Position
...

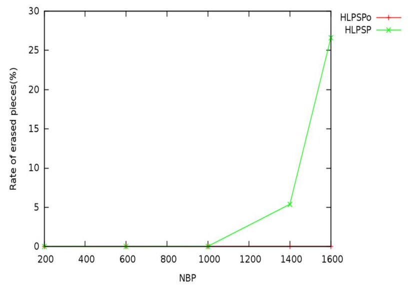
Fig. 5: Buffer structure of HLPSP

VI. Conclusion

In this paper, we raised the problem of erasing the not-yet-played pieces when a circular buffer is used to save the future pieces. We showed through simulation done in the context of the recently proposed P2P protocol called HLPSP that this phenomena affects the playback experience of the user. We proposed a pieces requesting control mechanism to eliminate the erasing phenomena. The simulation results showed that this mechanism led to decreasing the rate of the missed pieces



(a) Rate of missed pieces as a function of the start up buffering for HLPSP



(b) Rate of erased pieces as a function of the start up buffering for HLPSP

Fig. 6: Rate of missed pieces and erased pieces for a buffer size equal to 1000

during the playback which means a more comfortable playback experience for the user.

Currently, we are investigating 1) how to enhance the performances of HLPSP through implementing new pieces' selection algorithm and 2) how to secure the HLPSP network against the malicious nodes' attacks in a dynamic environment.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014," Jun. 2010.

[2] I. Lassoued, J.-M. Bonnin, and A. Belghith, "Towards an architecture for mobility management and resource control," in *Wireless Communications and Networking Conference, (IEEE WCNC'08)*. IEEE, 2008, pp. 2846–2851.

[3] C. Huang, J. Li, and K. W. Ross, "Can Internet Video-on-Demand be Profitable?" 2007.

[4] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, mar 2008.

[5] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," pp. –, 2008.

[6] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale p2p iptv system," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1672–1687, 2007.

[7] qqlive Team, "The qqlive official website," October 2009. [Online]. Available: http://www.qqlive.com/

[8] T. Team, "The tvant official website," October 2009. [Online]. Available: http://http://www.tvants.com/

[9] C. Hammami, I. Jemili, A. Gazdar, A. Belghith, and M. Mosbah, "Hybrid live p2p streaming protocol," *Procedia Computer Science*, vol. 32, no. Complete, pp. 158–165, 2014.

[10] ——, "HLPSP: A hybrid live P2P streaming protocol," *TIIS*, vol. 9, no. 3, pp. 1035–1056, 2015. [Online]. Available: http://dx.doi.org/10.3837/tiis.2015.03.011

[11] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast." pp. 205–217, 2002.

[12] D. Tran, K. Hua, and T. Do, "Zigzag: an efficient peer-to-peer scheme for media streaming," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, 2003, pp. 1283–1292 vol.2.

[13] A. Payberah, F. Rahimian, S. Haridi, and J. Dowling, "Sepidar: Incentivized market-based p2p live-streaming on the gradient overlay network," in *Multimedia (ISM), 2010 IEEE International Symposium on*, 2010, pp. 1–8.

[14] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in cooperative environments," in *Peer-to-Peer Systems II*. Springer, 2003, pp. 292–303.

[15] X. Zhang, J. Liu, B. Li, and T. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, 2005, pp. 2102–2111 vol. 3.

[16] S. Xie, B. Li, G. Keung, and X. Zhang, "Coolstreaming: Design, theory, and practice," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1661–1671, 2007.

[17] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1415–1423.

[18] F. Wang, Y. Xiong, and J. Liu, "mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast," pp. 49–49, 2007.

[19] K. Maalaoui, A. Belghith, J.-M. Bonnin, and M. Tezeghdanti, "Performance evaluation of qos routing algorithms," in *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*, 2005, p. 66.

[20] A. Belghith and M. A. Abid, "Autonomic self tunable proactive routing in mobile ad hoc networks," in *Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on*, October 2009, pp. 276–281.

[21] M. Belhassen, A. Belghith, and M. A. Abid, "Performance evaluation of a cartography enhanced olsr for mobile multi-hop ad hoc networks," in *Wireless Advanced (WiAd'11)*. IEEE, 2011, pp. 149–155.

[22] Q. Zhu, R. Wang, D. Qian, and F. Xiao, "Re-exploring the potential of using tree structure in p2p live streaming networks," pp. 125–132, 2009.

[23] S. Asaduzzaman, Y. Qiao, and G. Bochmann, "Cliquestream: Creating an efficient and resilient transport overlay for peer-to-peer live streaming using a clustered dht," *Peer-to-Peer Networking and Applications*, vol. 3, no. 2, pp. 100–114, 2010. [Online]. Available: http://dx.doi.org/10.1007/s12083-009-0052-8

[24] P. Team, "The ppstream official website," October 2013. [Online]. Available: http://www.ppstream.com/

[25] U. Team, "Uusee," October 2013. [Online]. Available: http://www.uusee.com/

[26] C. Feng and B. Li, "Understanding the performance gap between pull-based mesh streaming," University of Toronto, Tech. Rep., 2008.

[27] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1678–1694, 2007. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2007.071207

[28] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for p2p streaming systems." in *INFOCOM*. IEEE, 2007, pp. 919–927.

[29] A. Varga, "Using the omnet++ discrete event simulation system in education," *IEEE Trans. on Educ.*, vol. 42, no. 4, pp. 11 pp.–, Nov. 1999.

[30] "Gt-itm topologies for the omnet++ simulation platform and oversim framework [online]," 2010.

[31] Y. Lu, B. Fallica, F. A. Kuipers, R. E. Kooij, and P. V. Mieghem, "Assessing the quality of experience of sopcast," *Int. J. Internet Protoc. Technol.*, vol. 4, no. 1, pp. 11–23, Mar. 2009.