HLPSP: A Hybrid Live P2P Streaming Protocol

Chourouk Hammami[1], Imen Jemili[1], Achraf Gazdar[2], Abdelfettah Belghith[2] and Mohamed Mosbah[3]
[1] HANA Lab, ENSI, University of Manouba
Manouba 2010, Tunis, Tunisia
[e-mail: chourouk.ham@gmail.com, imen.jemili@hanalab.org]
[2] College of Computer and Information Sciences, King Saud University,
Riyadh 11543, Saudi Arabia,
[e-mail: abelghith@ksu.edu.sa , agazdar@ksu.edu.sa]
[3] LaBRI, University of Bordeaux,
Talence, France
[e-mail: mohamed.mosbah@labri.fr]
*Corresponding author: abdelfettah Belghith

Received August 11, 2014; revised December 21, 2014; accepted January 21, 2015; published March 31, 2015

Abstract

The efficiency of live Peer-to-Peer (P2P) streaming protocols depends on the appropriateness and the management abilities of their underlying overlay multicast. While a tree overlay structure confines transmission delays efficiently by maintaining deterministic delivery paths, an overlay mesh structure provides adequate resiliency to peers dynamics and easy maintenance. On the other hand, content freshness, playback fluidity and streaming continuity are still challenging issues that require viable solutions.
In this paper, we propose a Hybrid Live P2P Streaming Protocol (HLPSP) based on a hybrid overlay multicast that integrates the efficiency of both the tree and mesh structures. Extensive simulations using OMNET++ are conducted to investigate the efficiency of HLPSP in terms of relevant performance metrics, and position HLPSP with respect to DenaCast the enhanced version of the well-known CoolStreaming protocol. Simulation results show that HLPSP outperforms DenaCast in terms of startup delay, end-to-end delay, play-back delay and data loss.

Keywords: Live Streaming, Peer-to-Peer, Hybrid P2P Overlay

http://dx.doi.org/10.3837/tiis.2015.03.011 **ISSN : 1976-7277**

1. Introduction

With the widespread penetration of broadband accesses, multimedia services are getting increasingly popular among users and are contributing to a significant amount of today's Internet traffic. Live video applications in the Internet, involving live media streaming from a source to a large population of users, have become more and more popular. These delay-sensitive applications often require the collective use of massively distributed network resources and impose challenging constraints to satisfy robust and reliable support of media streaming. Therefore, such applications are not adequately supported by the traditional client-server architecture in the Internet.

Relying on an overlay multicast for media streaming over the Internet has been intensively studied. The overlay multicast can be classified into two broad categories [1]: infrastructure-based overlay multicast and peer-to-peer (P2P) overlay multicast.

The infrastructure-based overlay multicast deploys dedicated proxies at the edge of the Internet by the service provider, and constructs a collaborative service overlay network (SON). Due to the relative stability of proxy nodes, such a solution provides a robust and reliable media streaming for many commercial services [1]. Despite enabling a resilient market that can be easily controlled and managed by service providers, the cost issues remain a problem to be solved because of the deployment of a large number of servers throughout the Internet.

On the other hand, Peer-to-Peer (P2P) overlay networks enable efficient resource sharing in distributed environments and provide a highly effective and scalable solution to this problem. Such an approach eliminates the need for dedicated servers to mediate between end systems and exploits client resources to forward the media and to offer a flexible and scalable solution for live streaming. Currently, there are many commercially available P2P streaming and multicast video application systems [2-7], such as DONet/CoolStreaming [2], PPlive [7] and PPStream [6].

In this paper, we focus on P2P overlay networks. There are two widely adopted overlay approaches in P2P-based live streaming systems: the tree approach and the mesh approach. The tree-based protocols [8, 9] build a tree overlay on the application layer. It borrows its concepts from IP multicast where the media is pushed from the root to interior nodes towards the leaf nodes. The short latency of the data delivery is the main benefit of this approach. However, the tree structure is very fragile since a failure of a node close to the root would affect most of the traffic forwarded to descendant interior nodes. In fact, when any interior member leaves, the tree becomes disconnected and the children of this departing node need to be reconnected to the tree. Frequent tree reconnections at peer churns may entail a considerable control overhead and would greatly affect the system performance.

An alternative to tree structured overlays is the mesh structure [3, 10, 11, 28] in which the nodes are rather connected in a mesh. They use the exchanged availability of the data between partners to guide the data flow. Since peers can receive data from multiple peers and provide data to several others, the mesh structure is more reliable and is highly resilient to node failures than that of a tree. However, the mesh structure is subject to unpredictable latencies due to frequent exchanges of notifications.

The question naturally arises as to how we can design an overlay structure that confines transmission delays efficiently by maintaining deterministic delivery paths as in the tree structure, yet provides adequate resiliency and easy maintenance to peers' dynamics as in the mesh structure. In this paper, we propose a new design for the tree-based approach to remedy

its shortages while retaining its advantages. We provide a more resilient overlay to peers' dynamics by exploiting the benefits of mesh-based P2P overlays which outperform the P2P tree-based approaches in term of much better reliability and smaller maintenance overhead.

Our proposed approach is called Hybrid Live P2P Streaming Protocol (HLPSP). It organizes peers into clusters according to their uploading capacities (a mesh topology), moving consequently the powerful peers closer to the source. Each cluster represents a level of uploading capacity. A mesh topology connects the nodes within each cluster in the overlay. Similar to the tree topology, the multimedia contents are delivered from the source, representing the highest level (level 0) down to the lowest levels. Each peer maintains two sets: a set of partners from which to retrieve media content and another set of partners for which to serve and provide media content. As such, the HLPSP is based on a hybrid topology trying to insure a certain tradeoff between the tree and the mesh topologies by combining their benefits.

The rest of this paper is organized as follows: In section 2, we present related relevant works in this area. Section 3 presents HLPSP, its overlay construction and the communication between the different actors. The performance evaluation of HLPSP is presented in section 4. Section 5 concludes the paper and briefly discusses some future research orientations.

2. Related Work

Existing P2P overlay multicast approaches for live streaming can be roughly classified into two categories [12]: the tree-based overlay multicast and the mesh-based P2P overlays. The central idea in the tree-based approach is to construct a multicast tree among end hosts using an overlay network. The tree protocol organizes peers into a tree rooted at the source server. In this structure, nodes without children are called leaf nodes and internal nodes are called branch nodes. Streaming flows originating from the root go down along branch nodes until reaching the leaf nodes. NICE [8] and ZigZag [9] are examples of this kind of protocols. This single-tree based overlay suffers from two drawbacks:1) potentially a 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 a node close to the source can cause significant network disruption and requires a re-construction of the overlay topology. The multi-tree approach [5, 13] was introduced to tackle single-tree's problems. In such an approach, the source encodes the stream into several sub-streams and distributes each substream along a separate overlay tree. There are two key improvements achieved by the multi-tree solution. First, the overall system becomes more resilient since nodes in one tree are much less affected by failures of nodes on another tree. Second, the bandwidth of all the nodes can be more fairly utilized as long as each node can only be a leaf of just one tree. However, a multi-tree scheme is more complex to manage specially in the presence of high network dynamics. A further enhancement technique was proposed by [26], where the authors introduced an optimal resource sharing between nodes in the P2P network. Their idea consists on making leaf nodes forward the video segments to nodes experiencing difficulties while downloading data from a parent peer with limited downlink bandwidth in a tree-based P2P overlay. They also proposed an optimal distribution of the downlink of nodes in a mesh-based overlay.

To improve the stability of service, mesh-based protocols have been proposed where each peer can accept media data from multiple parents and may provide servics to multiple children. Meshes based on the Gossip protocol thrive to find fresh peers. As such, this structure is highly resilient to node failures, however it remains subject to unpredictable latencies caused by

frequent exchanges of notifications and requests. PPlive [7], DONet/Coolstreaming [2, 11] and Prime [10] are examples of mesh based systems. The PPLive software [7] implements two major application level protocols: a gossip-based protocol for peer management and channel discovery, and a P2P-based video distribution protocol for a high quality video streaming. When an end-user becomes a PPLive peer node, it sends out a query message to the PPLive channel server to obtain an updated channel list. After selecting one channel to watch, the peer sends out multiple query messages to some root servers to retrieve an online peer list for this channel. Peers are identified within the list by their IP addresses and port numbers. Upon receiving a peer list, the PPLive client sends out probes to peers on the list to find active peers for the channel of interest. Some active peers may also return their own peer lists in the quest of helping the initial peer to find more peers. Consequently, peers share video chunks with each other. DONet/Coolstreaming [2, 11] aims to create a robust overlay mesh topology able to cope with node dynamics. In fact, peers gossip with one another for content availability information. As a result, a peer can independently select neighboring node(s) without any prior-structure constraint. The content delivery is based on a swarm-like technique using pull operations. The Coolstreaming protocol offers a reasonable video playback quality, however, it exhibits a long initial start-up delay caused by its random peer selection process and the per block pulling operation, and a high failure rate in joining a video program.

To improve performance, some hybrid systems combine tree and mesh structures to construct a data delivery overlay such as [14-19, 27]. The hybrid-based protocol DenaCast [14] is an enhanced version of the famous CoolStreaming protocol. It changes the way the CoolStreaming constructs its overlay. Within DenaCast, the tracker keeps a list of active peers in the network. Each peer requests neighbors from the tracker which returns some random peers based on the number mentioned in the request message. When a peer receives the neighbour candidates list, it starts getting neighbours by JOIN_REQ, JOIN_RSP and JOIN_ACK messages. We note that DenaCast does not differentiate between peers capacity, that is to say each peer has a fixed number of neighbours. In addition, the neighbours sent by the tracker are chosen randomly which makes the maintaining of the neighbourhood relation obsolete. Most of the hybrid solutions select some special nodes (based on some performance criteria) to construct the tree-based overlay after which these nodes communicate with their neighbours using a mesh topology. In contrast, the key idea of Treebone [15] is to identify a set of stable nodes to construct a tree-based backbone, called mtreebone. Most of the streaming data is pushed through the treebone and eventually reaches the outskirts. They also design a set of overlay constructions and evolution algorithms to minimize the startup and the transmission delays. Hence, the performance of the overlay closely depends on a small set of backbone nodes. However, it poses a series of unique and critical design challenges, in particular, the identification of stable nodes and seamless data delivery using both push and pull methods. In [17, 18], the authors presented the design of a P2P media streaming platform, named CliqueStream, that exploits the properties of a clustered P2P overlay to achieve both the locality properties and the robustness. It relies on the existence of more stable and higher bandwidth nodes in the network to allow the construction of efficient delivery structures without causing too much overhead from churn. It elects one or more stable nodes of highest available bandwidth in each cluster and assigns a special relaying role to them. To maintain transmission efficiency, a content delivery tree is constructed among the stable nodes using the structure in the underlying routing substrate and the content is pushed through them. Less stable nodes within a given cluster participate then in the content dissemination and the pull of the content creating a mesh around the stable nodes. However, the overhead for clusterhead selection and maintenance still remains high. In [27], the authors proposed a hybrid P2P

overlay construction where the peers are divided into domains. A domain represents a set of nodes having the same behavior and are near to each other to increase the stability of the domain. The nodes within the same domain and the same local network are organized in a tree based overlay. A mesh network links the domains to each other. Each domain contains a special node which plays the role of the domain head, called sub-root. This node is selected based on its life duration and the amount of data delivered to other nodes within the same domain. However, details are missing on how to select the sub-root at runtime and how to select an alternative sub-root in case of departure.

Besides, some other works have proposed overlay-construction techniques aiming to minimize the P2P traffic within the ISP networks through the design of P2P ISP-friendly solutions [29] and to avoid the free-rider behaviour by introducing peer incentives [28]. In this paper, we are rather assuming a full cooperation of all nodes in the P2P network without recourse to incintives which could be integrated in future work.

While virtually all prior works rely mainly on a random peer selection scheme, our proposed HLPSP organizes the nodes on levels based on their uploading capacities. In HLPSP, the content flows consecutively from the source at the lowest level to higher levels according to the tree based topology while all the levels are interconnected according to the mesh based topology. In contrast to other hybrid solutions, HLPSP provides more than just one special node within the same level which increases continuity of downloading from the other nodes even if some nodes leave the network. The functional behaviour of HLPSP moves the powerful peers (in terms of the uploading capacity) close to the source which allows minimizing the end-to-end delay, and consequently maximizing the freshnes and serving the maximum numer of peers.

3. Hybrid Live P2P Streaming Protocol: HLPSP

HLPSP constructs its overlay based on the uploading capacities of the nodes. It enforces nodes with high capacities to be located closer to the source node. As such, a node with a limited upload capacity or bandwidth does not throttle the streaming and consequently does not degrade the system performance. Recall that P2P streaming is a sensitive delay application that requires stringent continuity.

The central idea introduced by HLPSP [20] is to organize the nodes into levels based on their uploading capacities as illustrated on **Fig. 1**. The source of the media, which is supposed to have a large enough uploading capacity, is the unique node that occupies level 0 which is the base (lowest) level. All the other nodes are distributed into levels according to their uploading capacities. Each overlay level represents a predefined range of bandwidth or uploading capacity. Nodes having high uploading capacities are assigned to lower levels which are closer to the source node. Levels assigned to nodes other than the source, vary from level **1** (the lowest level) to level **x** (the highest level), where **x>1**.

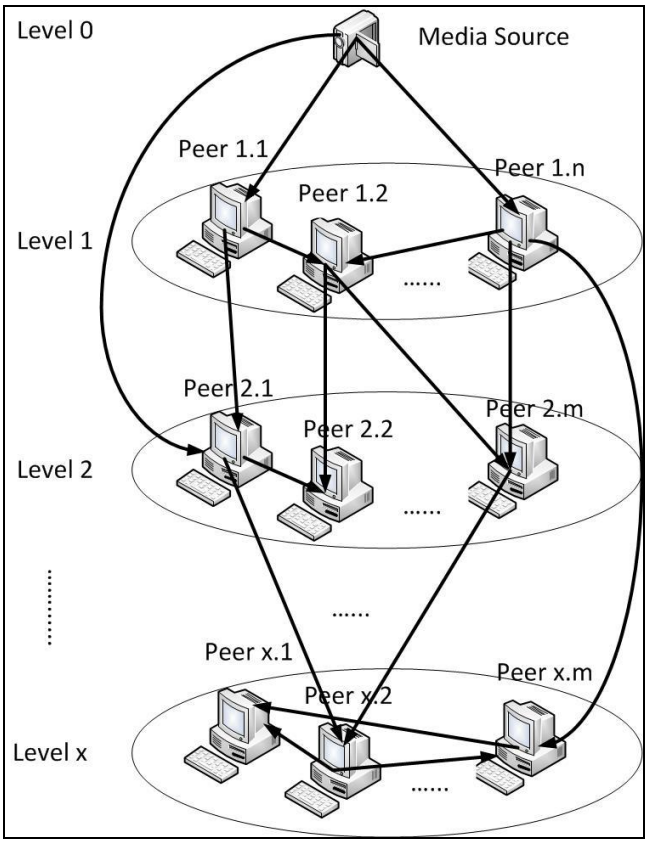


Fig. 1. HLPSP overlay

A node at level **y** (**1<= y <= x**) can be served by nodes, called **active nodes**, belonging to the same or a lower level **i (1<= i<= y**). The gathering of the media streaming from different active neighbors, is intended to increase the degree of cooperation which tacitely leads to a better playback continuity. A node at a given level **j** can serve nodes at the same level or at a higher level **k** (**j<= k <= x**), called **passive nodes**. Recall here that we do not consider the existence of selfish participants which might delay or throttle forwarding the streamed data. Such a behavior can affect the overall streaming quality. We rely on node full cooperation to insure the propagation of the streamed data in a relatively continuous way even with node dynamics.

A salient feature of HLPSP is to insert a new coming node at the appropriate level which contributes to enhance the overlay performance. In fact, the functional behavior of our approach allows nodes joining the system to integrate the overlay and to be inserted in the appropriate level taking into account their uploading capacities and without disturbing the on going streaming of other nodes. Besides, to facilitate the overlay construction and maintenance phases, we add some intelligence to the tracker. It is the tracker responsibility to find suitable active neighbors when receiving the request of a new joining node to the system. In this way, we avoid disagreements and delays caused when trying to contact overloaded nodes or non existant nodes that already departed from the system. As such, we are able to reduce the startup delay and manage more efficiently node departures.

3.1 Overlay construction

Three actors are involved in HLPSP: the media source, the tracker and the peers. Initially upon registering, the source sends its uploading capacity to the tracker. The later calculates the maximum number of passive peers for the source, denoted **NPmax**, according to the following formula:

$$NP\,max = UploadCapacity / PieceSize \quad (1)$$

where **UploadCapacity** refers to the node uploading capacity and **PieceSize** corresponds to the average size of exchanged pieces.

After achieving this step, the tracker starts accepting demands associated with this source streaming through neighbor requests sent by the peers. These requests are used in two cases:

- Upon joining the system: when a new node joins the system, it contacts the tracker in order to obtain a list of available nodes having enough capacity to serve new peers. The number of these active nodes in the returned list is denoted **NAcurrent**.
- During streaming: when the number of active nodes effectively serving a given peer, denoted by **NAeffective**, is less than the average **NAeffective** of this peer's level. Practically, not all **NAcurrent** active nodes are needed and used by the peer to download the required content pieces. In other words, **NAeffective** is always smaller than or equal to **NAcurrent.**

Upon the reception of a neighbor request from a peer, the tracker responds by a neighbor response which contains the list of potential available active neighbors.

Moreover, every peer sends periodically to the tracker an update message which contains the list of its current active neighbors and specifies the current effective neighbors selected for retrieving the requested pieces.

3.2 Neighbor request handling

Upon receiving a neighbor request from peer $\mathbf{p_i}$, the tracker verifies if this message is received for the first time. In such a case, it computes **NPmax** and **NAmax**, where **NPmax** refers to the maximum number of passive neighbors for this new peer according to equation 1, and **NAmax** refers to the maximum number of active neighbors for this new peer computed as in formula 2 below. Otherwise, the tracker just submits an updated list of active neighbors based on the already calculated values of **NAmax** and **NPmax**. The updating of this list is accomplished by the same selection process.

$$NA\max = DownloadCapacity / PieceSize \quad (2)$$

where **DownloadCapacity** refers to the requesting peer downloading capacity and **PieceSize** corresponds to the average size of exchanged pieces.

Once these two variables are calculated, the tracker assigns peer $\mathbf{p_i}$ to an appropriate level, based mainly on the number of its calculated passive neighbors (i.e.; its uploading capacity). A proper selection process, given formely by **Algorithm 1** below, is performed by the tracker to provide each newly joined node with a list of active nodes, denoted $\mathbf{Liste_{NA}}$. $\mathbf{Liste_{NA}}$ contains a list of triplets (p_i, type, p_k) where p_i is the identity of the peer requesting the list of active neighbors, type is either 1 or 2 depending on the type of the join_request to be executed by p_i, and p_k is either null or a peer identity in case of redirection. This will be detailed in **Section 3.3** below. The active nodes are designated from the lower levels starting from the source down to and including the level of peer $\mathbf{p_i}$. Nodes from the lower layers and the same layer that still

have sufficient remaining uploading capacities able to serve the new joining node, are designated as potential active nodes to within a predefined threshold.

The use of multiple active neighbors provides a high system resilience and reliability as the departure of one or few active neighbors does not impact the streaming continuity. Algorithm1 details the HLPSP selection process.

First of all, the tracker computes **NAcurrent** the size limit of **Liste$_{NA}$** which is equal to the minimum between **NAmax** of the requesting peer and a predefined threshold **th$_1$** set to 10 (this value was choosen based on several conducted simulation experiments). This limitation of the upper size of **Liste$_{NA}$** to **th$_1$** is due to the state of the art of current Internet access technologies, such as 4G and next coming 5G, which provide high access bandwidth. Formulas 1 and 2 give then large numbers of respectively passive neighbors **NPmax** and acive neighbors **NAmax**. This large number of passive peers (that in turn would be seen as a host of active neighbors by peers of higher levels) is neither needed in practice nor beneficial as it affects negatively the overlay performance. On the other hand, for an already connected peer asking for **Liste$_{NA}$** updating, the tracker will include a number of active neighbors equal to the the average **NAeffective** of the requesting peer's level, but greater or equal to a second predefined threshold denoted by **th$_2$.** The initial value of **th$_2$** is set to 2 according to conducted simulation measurements.

Secondly, the tracker selects the set of active neighbors to deliver to the requesting peer **p$_i$**. This is done by parsing levels starting from level **0** until the level of **p$_i$**. This parsing done in this order supplies active neighbors closer to the source which would garantee better performance as it will be investigated and shown later. At each level, the tracker tries looking for peers having **NPmax** > **NPcurrent**, where **NPcurrent** represents the current number of passive neighbors being actually served. The tracker stops the parsing whenever **Liste$_{NA}$** attains its upper size **NAcurrent** as described earlier. However, if after finishing the complete parsing, **Liste$_{NA}$** still contains less than **th$_2$** active neighbors then the tracker launches an overlay redirection procedure for the account of peer **p$_i$**. This overlay redirection or patching procedure consists in finding a peer **p$_j$** on the same level as the new coming peer **p$_i$** or on a lower level that serves a passive node **p$_k$** at a higher level than that of the level computed by the tracker for **p$_i$**. This search is performed in the order starting from the level of requesting peer **p$_i$** and backwards to the source. If such peers **(**the passive peer **p$_k$** and its active peer **p$_j$**) are found, then a redirection will be performed; meaning that **p$_k$** will be disconnected from its active peer **p$_j$** and will be reconnected to **p$_i$** instead. As for **p$_i$**, it will connect to **p$_j$** as an additional active peer and of course p_i will serve **p$_k$** as one of its passive peer. This will be repeated as long as needed to attain a number of active peers greater or equal to **th$_2$** (the second part of **Algorithm 1** lines 14-27).

Algorithm 1 **Peer Selection Process**
/* Building **Liste$_{NA}$** upon receiving a neighbor request from a new peer p_i */

Require: Peer identity p_i, level(p_i), NAcurrent(p_i), th_2
Ensure: Fill the Liste$_{NA}$((p_i , type, peer id))
1: lev ← 0
2: /* search for active neighbors that will serve peer p_i*/
3: **while** lev <= level(p_i) and sizeof (Liste$_{NA}$(p_i)) < NAcurrent(p_i) **do**
4: Let CANDIDATE be the set of peers of level lev having
5: NAeffective <= NAcurrent
6: **repeat**

7: Let p_j in CANDIDATE
8: Remove p_j from CANDIDATE
9: Add (p_j, type 1, null) to $Liste_{NA}(p_i)$
10: Update sizeof ($Liste_{NA}(p_i)$)
11: **until** CANDIADTE=∅ or sizeof ($Liste_{NA}(p_i)$) = NAcurrent(p_i)
12: lev ← lev+1
13: **end while**
14: /* redirection procedure for the account of peer p_i*/
15: lev ← level(p_i)
16: **while** lev >= 0 and sizeof ($Liste_{NA}(p_i)$) < th_2 **do**
17: Let CANDIDATE be the set of peers at level lev other than p_i having passive neighbors belonging to levels lower than lev
18: Let REDIRECTED= ∅
19: **repeat**
20: Let p_j in CANDIDATE and let p_k one of its passive neighbors at level lower than that of p_i and not in REDIRECTED
21: Remove p_j from CANDIDATE
22: Add p_k to REDIRECTED
23: Add (p_j, type 2, p_k) to $Liste_{NA}(p_i)$
24: Update sizeof ($Liste_{NA}(p_i)$)
25: **until** CANDIADTE=∅ or sizeof ($Liste_{NA}(p_i)$) = th_2
26: lev ← lev-1
27: **end while**

3.3 Tracker response handling

Once peer $\mathbf{p_i}$ receives the tracker response (i.e.; the list of triplets $\mathbf{Liste_{NA}}$), it starts establishing its neighborhood relationships. Three types of messages can be exchanged between coordinating peers: the join_request, the join_response and the join_deny depending on the type found in each triplet in $\mathbf{Liste_{NA}}$. A join_request is sent by $\mathbf{p_i}$ to every active neighbor in the $\mathbf{Liste_{NA}}$ provided by the tracker. We can distinguish between two types of join-request messages:

1. **Direct join_request (Type 1):** peer $\mathbf{p_i}$ sends a join_request (type 1) to each peer tagged as type 1 in $\mathbf{Liste_{NA}}$ in the quest to be connected as a passive node. Each of the solicited neighbors responds $\mathbf{p_i}$ with a join_response accepting being its new active neighbor. This case is illustrated through the example portrayed on **Fig. 2**. The source node is always a supplier, meaning that it can only have passive neighbors. In the example, the source can supports up to 6 passive peers. Just under each peer, a triplet indicates the level of the peer, its NAmax and its NPmax. A newcomer **P6** has been assigned level 1 by the tracker according to its uploading capacity. $\mathbf{Liste_{NA}}$(P6), the list of active neighbors provided by the tracker to peer P6, should contain the following triplets: (source, type 1, null) (P1, type 1, null). Peer P6 sends a join_request to the source and another one to P1. Peer P6 has currently NAeffective=2, NPmax=3 and NAmax=3. The source and peer P1 respond P6 positively using join_response and the connections will be set up. Recall, we are here using th_2=2.

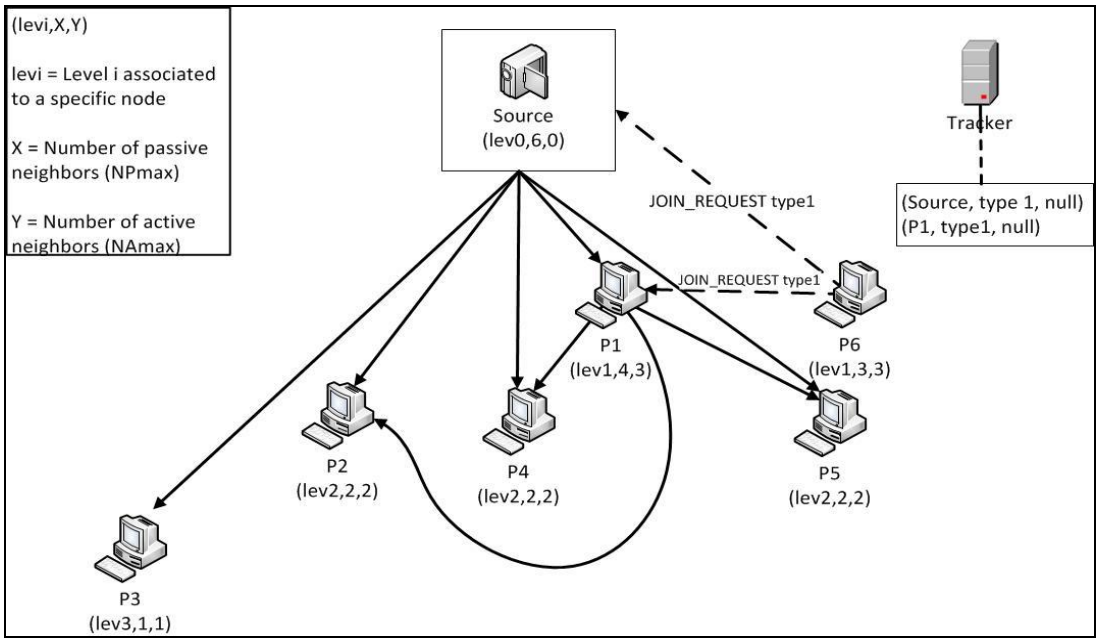


Fig. 2. Case of Direct join request

2. **Redirection request (Type 2):** One of the main innovations introduced by our approach is the concept of **redirection** allowing a new coming $\mathbf{p_i}$ to be inserted in the appropriate place in a way to harness its uploading capacity suitably. This case arises when the tracker in its scan from the source up to the level of the requesting peer $\mathbf{p_i}$ cannot find enough active peers to serve $\mathbf{p_i}$ (line 6 of **Algorithm 1**); that is the constructed $\mathbf{Liste_{NA}(p_i)}$ contains less than $\mathbf{th_1}$ entries. Consequently, the tracker launches a scan starting from $\mathbf{p_i}$'s level down to the source in the quest of an active neighbor candidate $\mathbf{p_j}$ (which currently has no more room to accept more passive neighbors) having one of its current passive neighbors, say $\mathbf{p_k}$, assigned to a greater level (meaning $\mathbf{p_k}$ has a lower uploading capacity than $\mathbf{p_i}$) than the requesting $\mathbf{p_i}$. The idea is to prune the relation (the connection) between $\mathbf{p_j}$ and $\mathbf{p_k}$ meaning that $\mathbf{p_j}$ is no more an active neighbor for $\mathbf{p_k}$, establish $\mathbf{p_j}$ as an active neighbor of $\mathbf{p_i}$, and establish $\mathbf{p_i}$ as an active neighbor of $\mathbf{p_k}$. This is the redirection procedure that redirects peer $\mathbf{p_k}$ to be served by $\mathbf{p_i}$ instead of $\mathbf{p_j}$ and provides **pi** with $\mathbf{p_j}$ as an additional active neighbor. The rationale behind this is twofold. First, this enforces $\mathbf{th_2}$ to guarantee fluidity and continuity of the streaming. Second, peer $\mathbf{p_i}$ has a higher uploading capacity than $\mathbf{p_k}$ and therefore avoids the potential traffic throttling or bottleneck that might be caused by $\mathbf{p_k}$. Upon receiving a join request of type 2 from $\mathbf{p_i}$, peer $\mathbf{p_j}$ sends a join deny to its passive peer $\mathbf{p_k}$ and encloses the address of $\mathbf{p_i}$ within the deny message. Thanks to this information, passive peer $\mathbf{p_k}$ sends a join_request to its new active neighbor peer $\mathbf{p_i}$. Finally, peer $\mathbf{p_j}$ sends a join_response to $\mathbf{p_i}$ accepting being its new active neighbor. It should be noticed that a passive peer such as $\mathbf{p_k}$ cannot be involved in another additional redirection for the count of the same peer $\mathbf{p_i}$ otherwise it will be deprived from an active peer for each additional redirection. This is enforced by using the set REDIRECTED in **Algorithm 1**.

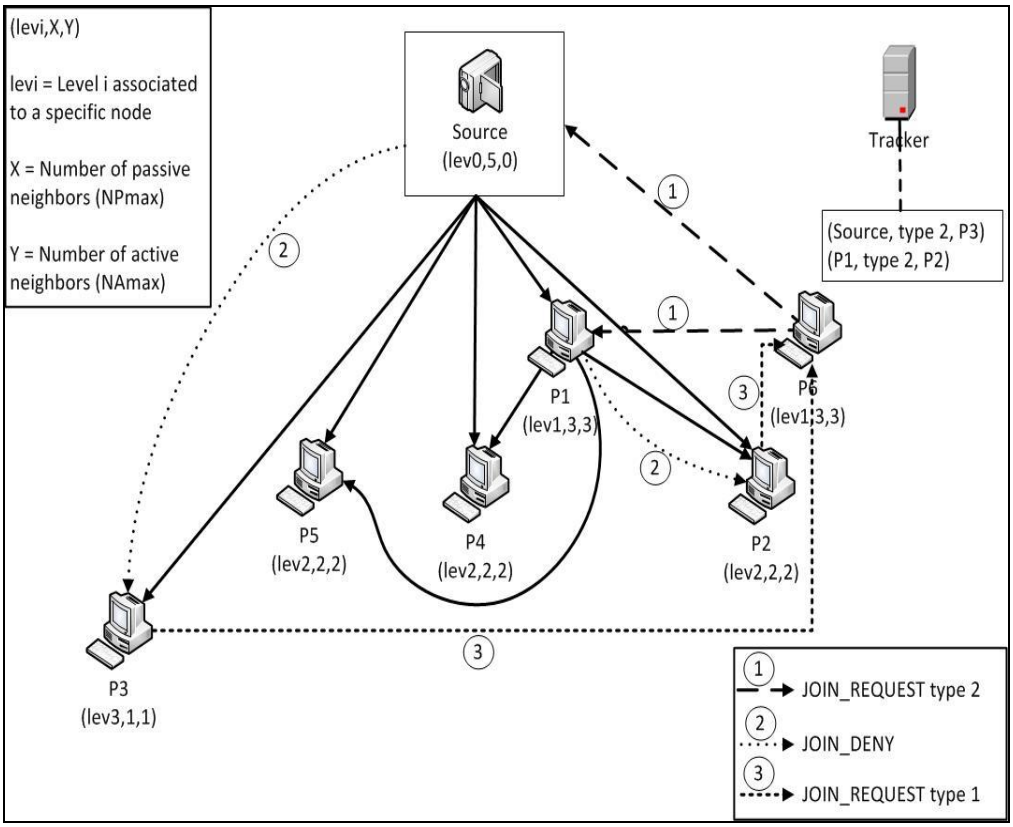


Fig. 3. a. Case of Redirection request

The example, illustrated in **Fig. 3. a**, exhibits this situation where peer **P6**, assigned to level 1, tries to integrate the overlay. As $\mathbf{th_2}$ is set to 2, we need to find at least two active neighbors to serve P6. Unfortunately the source at level 0 and P1 at level 1 cannot accept to serve P6 or any additional passive peer. A redirection is therefore required. The tracker through its selection process (**Algorithm 1** lines 14-27) **detects** that **P2** being at levels 2 is connected to P1 which is at the same level as P6. As such, P2 and its active peer P1 are a first case of redirection. Then it detects that P3 at level 3 is connected to the source, and therefore constitutes a second case of redirection. As a result the tracker constitutes the $\mathbf{Liste_{NA}(P6)}$ to contain the two triplets (P1, type 2, P2) and (source, type 2, P3) as shown on **Fig. 3. a**. **Fig. 3. b** portrays the establishment of the overlay after the execution of the two redirections.

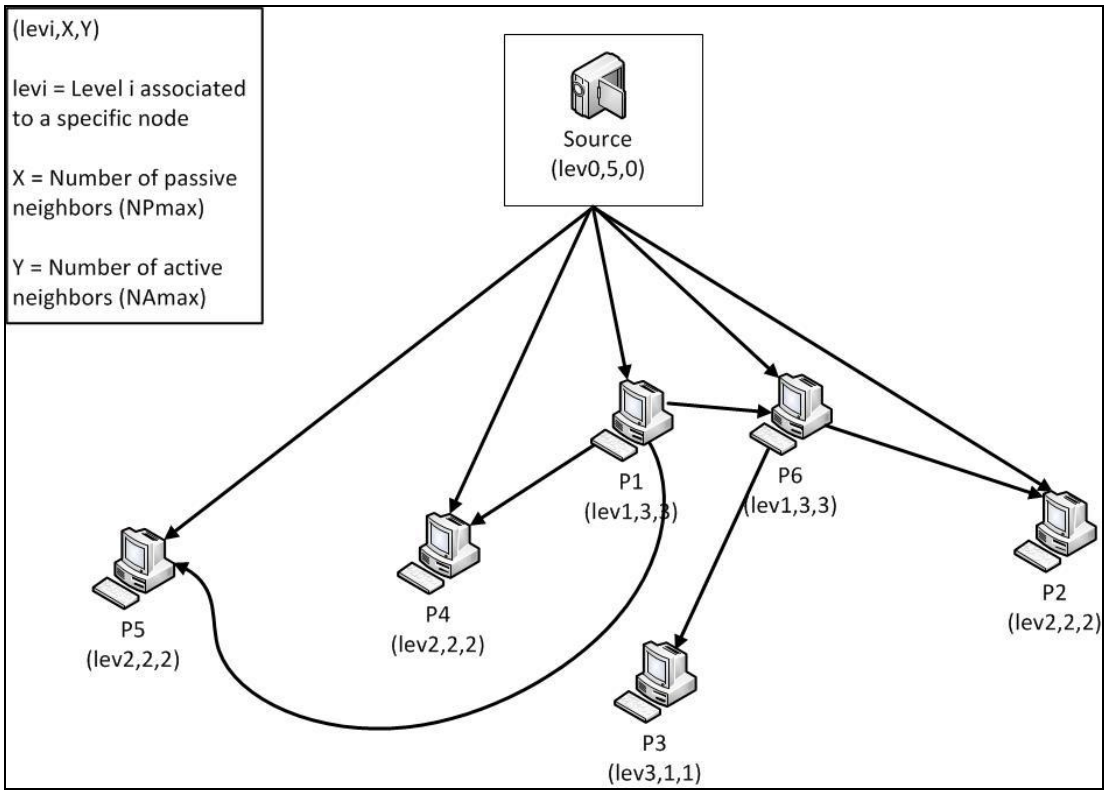


Fig. 3. b. Case of Redirection request

Periodically (each second in our conducted simulations), each peer sends to its passive neighbours a buffer map message containing the list of the media pieces or segments available in its own buffer. Recall that a video stream is usually divided into segments not necessarilly of the same length, and that the availability of the segments in the buffer of a node can be represented by a Buffer Map (BM). Based on the segment availability information periodically exchanged between nodes and their partners, each node is able to detect and schedule which segment is to be fetched from which partner. Two messages are exchanged between peers to download the media pieces:

1. Chunk_request is sent by a peer to an active neighbor to ask for a data piece,
2. Chunk_response is the response to the chunk_request and contains the requested piece.

Note here that we are assuming and adopting the same piece and peer selection algorithms as in DenaCast. First, the requester peer calculates the number of potential suppliers for each segment (i.e., the active neighbors containing in their buffers the target segment) and then chooses the one with the highest bandwidth and enough available time. In this work, we are solely focusing on the enhancements achieved by overlay construction as P2P streaming concentrates rather on the efficient delivery of audio and video content under tight timing constraints. Other different HLPSP specific piece and peer selection schemes than those used in DenaCast could surely be envisioned which may constitute a future orientation to augment this current work.

Finally, to accommodate overlay dynamics, each node sends periodically a message to the tracker announcing its existence and indicating the list of its effective active neighbors. BM

messages are also considered by passive peers as keepalive messages to be able to detect node departures.

4. HLPSP Performance Evaluation

To investigate the performance of our proposed scheme in different scenarios, we conducted a simulation based study using the discrete event network simulator, the OMNET++ simulator [21]. The network infrastructure used in the experiments is generated by the GT-ITM module [22]. DenaCast [14] is already implemented in the selected simulation environment. We investigate and position the behavior of HLPSP to that of DenaCast [14] when varying the number of nodes for static and dynamic P2P networks. We also investigate the impact of peer departures on the performance of our HLPSP approache only as DenaCAst doen't implement any departure handling mechanism.

The main rational behind our choice of comparing our proposed approach to DenaCast lies in the fact that DenaCast is a major representative of hybrid protocols as it represents an enhanced version of the famous CoolStreaming [2, 11] system. Furthermore, all other relevant hybrid protocols presented previously in the section 2, lack several technical details to be implemented in an appropriate way for comparison.

We used use Star Wars IV trace file to simulate a valid video stream which can be obtained from the Video Trace Library [23]. The streaming rate is set to 512 Kbps. The video stream is decomposed into several pieces having each an average size equal to 130 KB. Peers start playing (visualizing) the streamed media after a buffering of 40 seconds. This is comparable with the most widely deployed P2P live streaming system such as Sop-Cast's that has an average startup time of 30-45 seconds [24]. For all our experiments, we considered only one media source with an upload bandwidth of 5 Mbps.

The different peers uploding and downloading capacities are chosen so that they reflect the current deployed last mile access technologies. In fact, most peers are home users using the ADSL technology which is currently the prevalent deployed technology for Internet access from homes. P2P live streaming users are mostly individuals using essentially basic ADSL. Higher capacities offers are usually adopted by professionals and companies rather than induviduals at home. Taking this into account and in the quest to determine and position the performance of our HLPSP within a realistic set up, we adopted the following proportions: 10% (download 24 mbits, upload 2mbits), 10% (download 20 mbits, upload 1.5 mbits), 10% (download 12 mbits, upload 1.3 mbits), 15% (download 8 mbits, upload 1.2 mbits), 25% (download 4 mbits, upload 1.1 mbits), 35% (download 2 mbits, upload 1 mbits). **Table 1** summarizes our simulation parameters.

Table 1. Simulation parameters

Parameter	Value
Maximum packet size	1000 Bytes
Peer buffer	40 s
Buffermap exchange period	1 s
Selected trace file	Star Wars IV
Video codec	MPEG4 Part I
Video Frames Per Second (FPS)	25

Number of frames in a GoP	12 frames
Average piece size	130 Kb
Average video bit rate	512 Kbps
Number of sources	1
Capacity of peers	10% (down 24 mbits, up 2mbits) 10% (down 20 mbits, up 1.5 mbits) 10% (down 12 mbits, up 1.3 mbits) 15% (down 8 mbits, up 1.2 mbits) 25% (down 4 mbits, up 1.1 mbits) 35% (down 2 mbits, up 1 mbits)
Threshold th1	10
Threshold th_2	2
Number of levels	6
Simulation duration	200 s
Number of runs	10

4.1 Performance metrics

To evaluate the performances of HLPSP, we focus on the following performance metrics:

1. **Data Loss:** defined as the percentage of video content that is lost to the original video content. Formely, it is given by:

   Data Loss= (number of unreceived pieces / total number of pieces) × 10 (3)

2. **End-to-End delay :** Let T represents the time elapsing from the instant of creation of a piece at the source and the instant of its reception at a destination peer. The End-to-End delay is then defined as the average of T over all peers and all pieces composing the streamed video. Formely, the End-to-End delay is given by:

   $$End\ to\ end\ delay = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{\sum_{j=1}^{m}\left(tr_{j,k} - tc_j\right)}{m}\right) \quad (4)$$

   where n represents the number of peers, m corresponds to the number of pieces composing the streamed video, $tr_{j,k}$ denotes the reception instant of piece j by peer k and tc_j represents the creation instant of piece j at the source.

3. **Playback delay:** represents the average delay between the start instant of the playback by peers and the streaming time on the server. Formely, it is given by:

   $$Playback\ delay = \frac{1}{n}\sum_{m=1}^{n}\left(\frac{\sum_{j=1}^{m}\left(pt_{j,k} - tc_j\right)}{m}\right) \quad (5)$$

   where n represents the number of peers, m corresponds to the number of pieces composing the streamed video, $pt_{j,k}$ denotes the start of the playback instant of piece j by peer k, and tc_j represents the creation time of piece j at the source.

4. **Startup delay:** corresponds to the average delay between the connection instant to the overlay and the receiving instant of the first buffer map, over all peers. Formely, it is given by:

$$Startup\ \ delay = \frac{1}{n}\sum_{k=1}^{n}(trbm_k - tc_k) \quad (6)$$

where n represents the number of peers, trbm_k denotes the reception instant of the first buffermap by peer k and tc_k represents the connection instant of peer k.

5. **Connection overhead:** accounts for the required control messages exchanged among the peers and between the peers and the tracker during connection phases to the overlay. We purpusely ignored the signaling overhead caused by the exchange of the buffer maps.

4.2 Simulation results

We assume a peer arrival process following a that follows a uniform distribution with an inter-arrival time between 15 and 20 milliseconds. **Fig. 4** represents the data loss percentage as a function of the number of peers. We clearly observe the sensitivity of DenaCast to the ADSL technology. In DenaCast, the suppliers of pieces are selected randomly, while in HLPSP, the tracker provides each peer with a list of active neighbors selected smartly; namely active neighbors closer to the source. HLPSP provides 3 times more fluidity than DenaCast.

The startup delay in DenaCast is higher than that required by our HLPSP protocol as shown in **Fig. 5**. Recall that DenaCast selects neighbors randomly. In HLPSP, the tracker selects the active neighbors for a new arriving based on its actualized global view of the overlay. As such, the new arriving peer is guaranteed the confirmation of its join requests to these active peers. Moreover, in HLPSP peers periodically communicate to the tracker their effective active peers which allows the tracker to maintain a highly refreshed view of the overlay. In case of a peer departure, the traker upon a peer request finds and replace quickly and adequqtely the departed peer. We may notice on **Fig. 5**. for both protocols the decrease of the startup delay as the numbers of nodes becomes quit high. In this case of dense overlay, the tracker has more available active peers and consequently the choice of neighbors becomes more flexible.

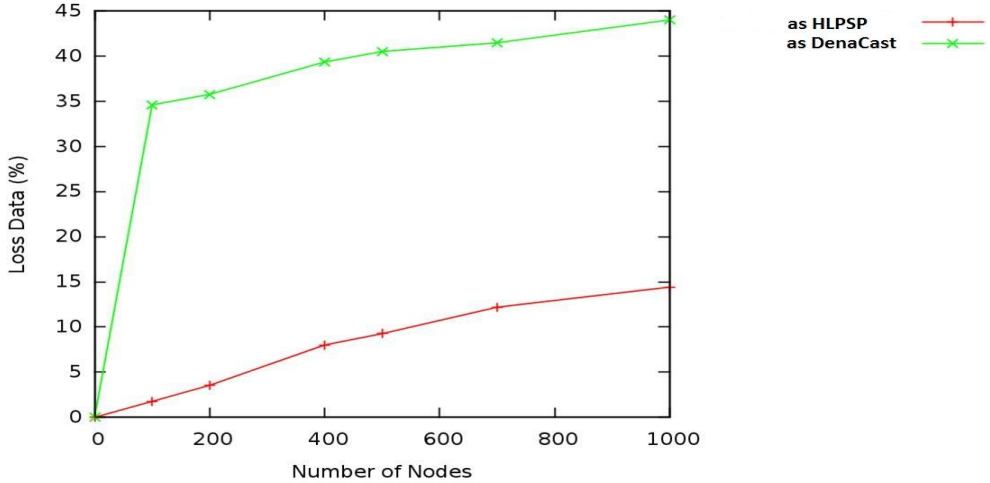


Fig. 4. The percentage of data loss as a function of the number of peers

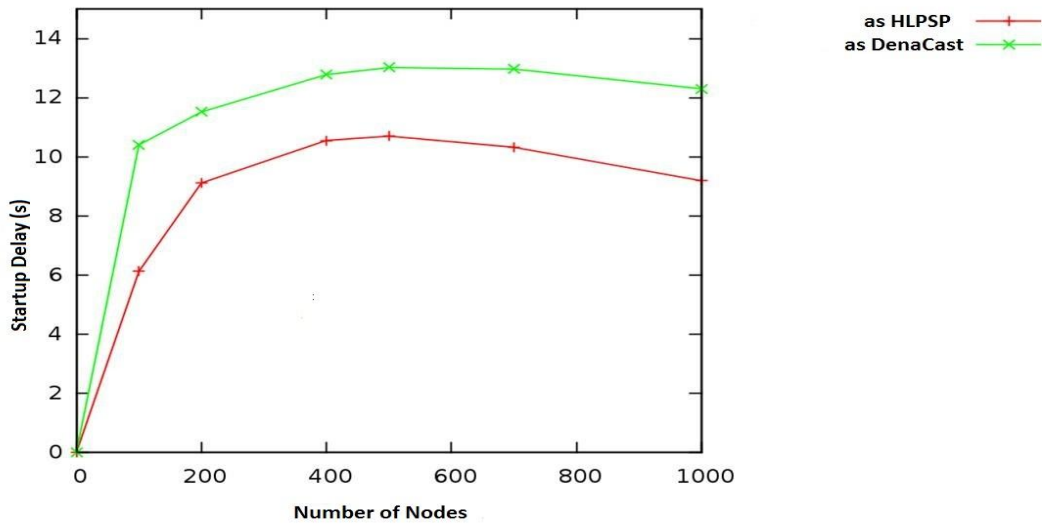


Fig. 5. The Startup delay as a function of the nodes' number

As we target live media streaming, freshness and continuity stand as primary constraints that need to be guaranteed. **Fig. 6**. portrays the end-to-end delay as a function of the number nodes for both protocols. We readily observe that data reception by peers is faster in HLPSP than in DenaCast due to the higher download capabilities in HLPSP. Many factors contribute to have a higher download speed of pieces in HLPSP than in DenaCast. First of all, the number of active neighbors of a peer is proportional to its download capacity, while for Denacast, all peers support the same number of partners without any consideration of their capacities. Second, in HLPSP, peers are supplied by active neighbors which are closer to the source. In addition, the redirection concept integrated in HLPSP allows peers to be placed in the appropriate levels and to exploit all their uploading capacities in serving other farther from the source. Moreover, in DenaCast, a peer requests and serves the pieces from and to a fixed number of peers selected randomly.

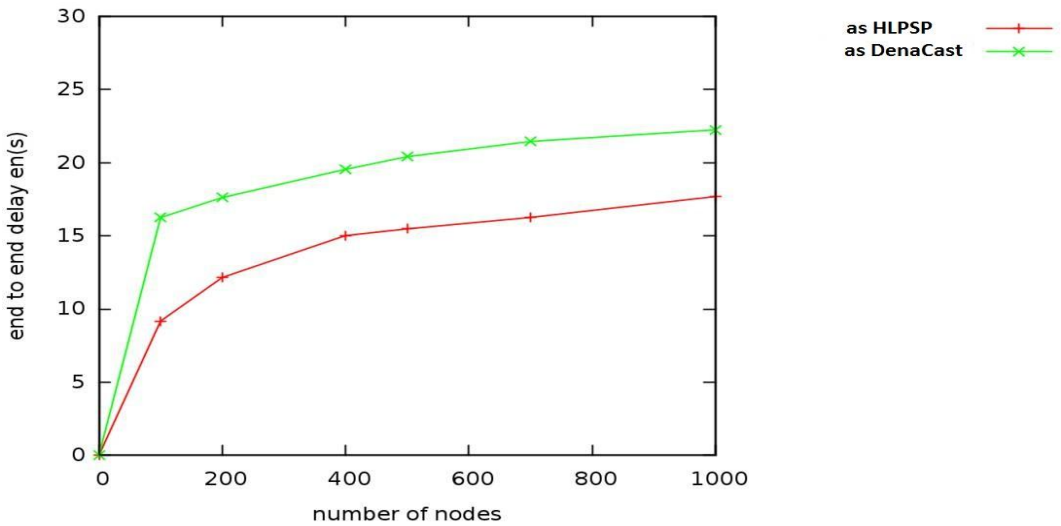


Fig. 6. The End-to-End delay as a function of the number of nodes

These results are consolidated by the simulation results of the Playback delay as a function of number of nodes as portrayed on **Fig. 7**. In HLPSP, peers experience a lesser playback delay. Selecting the appropriate active neighbors and placing nodes having high capacities close to the source yield much a better performance in terms of received video quality.

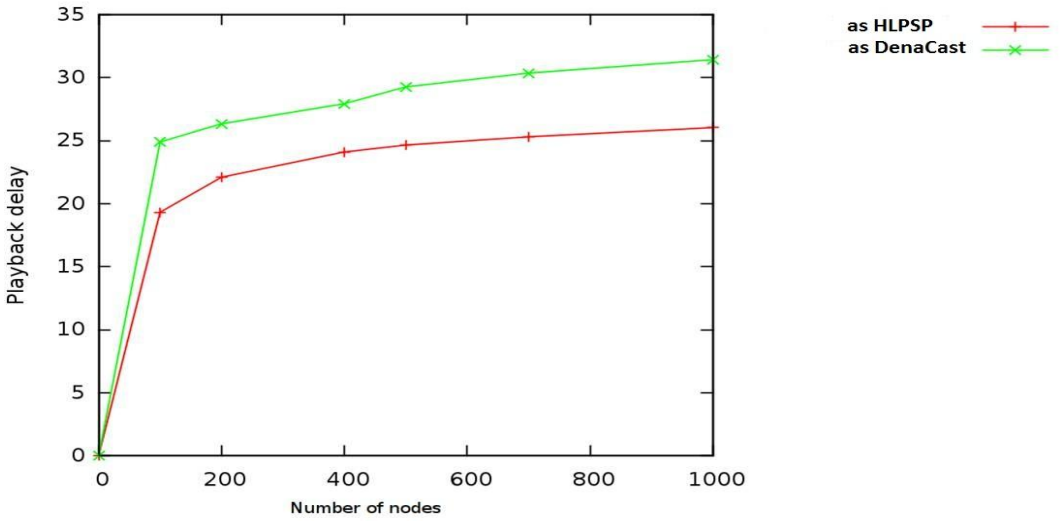
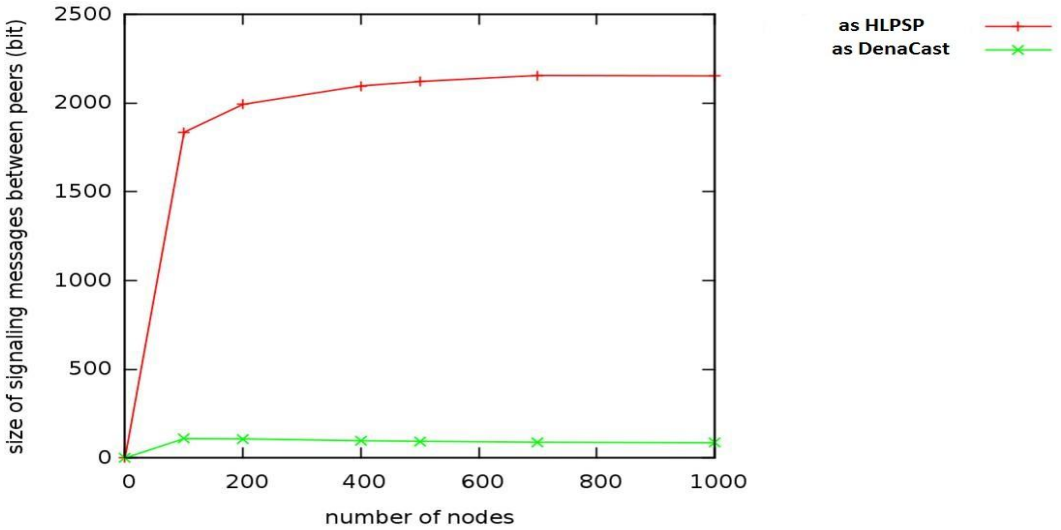
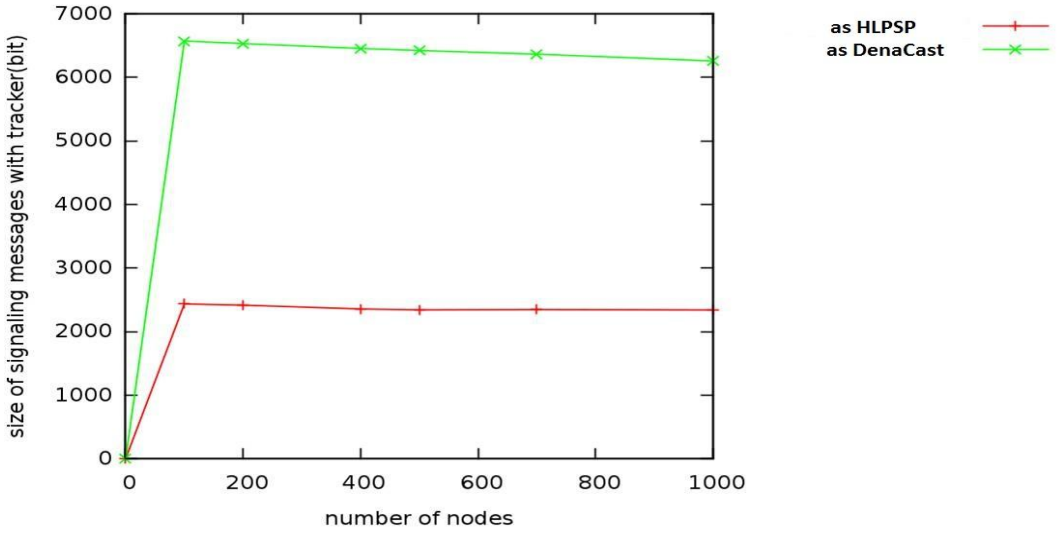


Fig. 7. The Playback delay as a function of the number of nodes

We now focus on the overhead associated with membership management. **Fig. 8. a** and **Fig. 8. b** illustrate respectively the amount of signaling exchanged among the peers and that between the peers and the tracker.



(a) Amount of signaling exchanged among the peers



(b) Amount of signaling exchanged between the peers and the tracker

Fig. 8. Amount of signaling exchanged within the HLPSP and the DenaCast networks

As expected, DenaCast requires less signaling among the peers than HLPSP since the later incorporates a much more involved peer placement and overlay updating which constitutes indeed one of its strength that allowed to accomplish better performances in terms of end-to-end delay, startup delay, playback delay and data loss. HLPSP not only inserts each new coming peer in the existing overlay at an the appropriate place, but also maintains an up to date view of the overlay which allows redirection and substitution of passive neighbors. The required additional join_request, join_response and join_deny messages naturally increase the control overhead.

However as portrayed on **Fig. 8. b**, DenaCast exercices more signaling between the peers and the tracker. In DenaCast partners are chosen randomly, which may provides the new peer with a list including already departed or overloaded nodes. In such a case, join_request messages will either be denied or not answered which will force the new peer to contact again the tracker to obtain a new list. The tracker in HLPSP provides instead a list of potential active neighbors able really to serve.

Now to ascertain the efficiency of our HLPSP protocol in a dynamic environment, we investigate the effect of the variation in the departure period of peers on the fluidity of the streamed video. We assume an overlay composed of 1000 nodes and we re-compute the data loss when 40% of the nodes leave the network at different simulation periods. We consider the folowing periods: uniform (30,50), uniform(30,80), uniform(30,110), uniform(30,150) and uniform(30,180). Recall that we have a simulation time of 200 seconds. **Fig. 9** represents the percentage of data loss as a function of the departure period of nodes. We do not consider DenaCast in **Fig. 9** as it lacks completely a mechanism to handle the departure of nodes. We clearly observe that HLPSP remains quite resilient to departures even when they are concentrated within the first half of the simulation period. Recall that we assumed a 40% departures, a rather high rate of departure. As these departures, despite their high rate, get more spread out over the simulation time, the percentage of data loss gets much better and approaches that of a complete static environment.

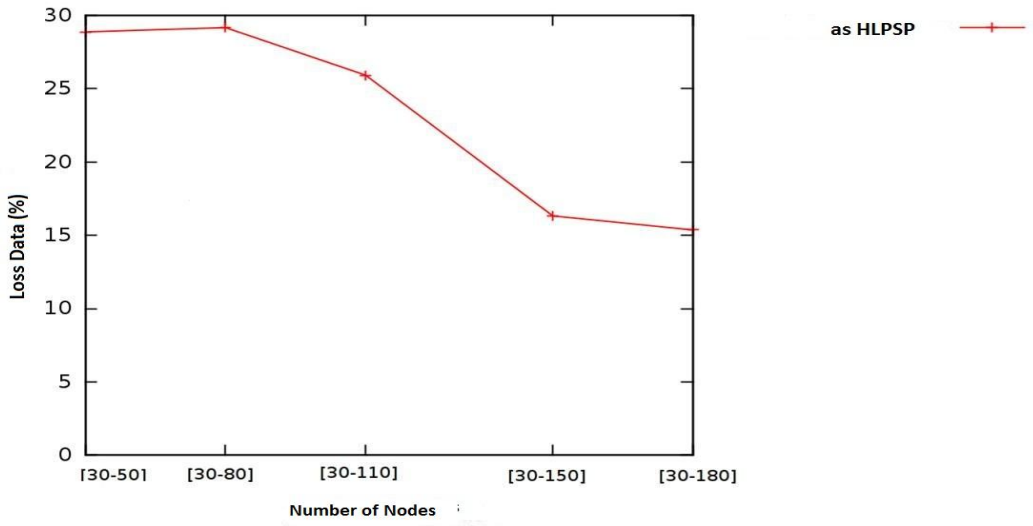


Fig. 9. The percentage of data loss as a function of the departure time of nodes

5. Conclusion

In this paper, we proposed a new design for the P2P overlay construction based on hybrid architecture (tree and mesh), called HLPSP. Within HLPSP, a peer requests pieces from many parents according to its download capacity. Later, it serves these pieces to many children corresponding to its upload capacity. We allow each newcomer to be inserted in the overlay in the appropriate place based on a list of active neighbors, selected smartly by the tracker.

Extensive simulations are conducted to evaluate and compare HLPSP against DenaCast, the enhanced version of CoolStreaming system. In particular, we showed through different scenarios that our proposal nicely outperforms DenaCast in terms of a better data fluidity, a better data freshness, a much lower of startup delay and a lower signaling overhead while communicating with the tracker. However the amount of signaling exchanged among the DenaCast peers is less than those of HLPSP. We believe that this is reasonable cost to pay with regards to the overall HLPSP performances.

Further improvements might concern specific piece and peer selection procedures. Moreover, streaming applications over P2P systems are deeply concerned with security; namely protection and privacy aspects [25]. Security aspects specific to HLPSP against malicious attacks, and integration of proper incentives are viable directions for further improvements.

Acknowledgment

This work was supported by the Research Center of the College of Computer and Information Sciences, King Saud University.

References

[1] B. Li, H. Yin, “Peer-to-peer live video streaming on the internet: issues, existing approaches, and challenges [peer-to-peer multimedia streaming],” *IEEE Communications Magazine*, vol. 45, no. 6, pp. 94-99. IEEE Press Piscataway, Jun. 2007. Article (CrossRef Link)

[2] X. Zhang, J. Liu, B. Li, T. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *Proc. of the 24th Annual Joint Conference Of the IEEE International Conference on Computer Communications (INFOCOM 2005)*. Proceedings IEEE, vol. 3, pp. 2102–2111, 2005. Article (CrossRef Link)

[3] D. Purandare, R. Guha, "An alliance based peering scheme for peer-to-peer live media streaming," *in Proc. of the 2007 workshop on Peer-to-peer streaming and IP-TV, P2P-TV '07*, ACM, pp. 340-345, 2007. Article (CrossRef Link)

[4] J. J. D. Mol, A. Bakker, J. Pouwelse, D. H. J. Epema, H. Sips, "The design and deployment of a bittorrent live video streaming solution," in *Proc. of the 11th IEEE International Symposium on Multimedia (ISM '09)*, IEEE Computer Society, pp. 342-349, 2009. Article (CrossRef Link)

[5] M. Castro, P. Druschel, A. marie Kermarrec, A. Nandi, A. Rowstron, A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," *In Proc. of the nineteenth ACM symposium on Operating systems principles (SOSP'03),* vol. 37, no. 5, pp. 298-313, Dec. 2003. Article (CrossRef Link)

[6] P. Team, The ppstream official website (Oct. 2013). URL http://www.ppstream.com/

[7] X. Hei, C. Liang, J. Liang, Y. Liu, K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia,* vol. 9, no. 8, pp. 1672–1687, 2007. Article (CrossRef Link)

[8] S. Banerjee, B. Bhattacharjee, C. Kommareddy, "Scalable application layer multicast," in *Proc. of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (*SIGCOMM'02)*,* vol. 32, no. 4, pp. 205-217, October 2002. Article (CrossRef Link)

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

[10] N. Magharei, R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," in *Proc. of the 26th IEEE International Conference on Computer Communications (INFOCOM' 2007)*. IEEE, pp. 1415–1423, 2007. Article (CrossRef Link)

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

[12] Yong Liu, Yang Guo, Chao 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. Article (CrossRef Link)

[13] A. Payberah, F. Rahimian, S. Haridi, J. Dowling, "Sepidar: Incentivized market-based p2p live-streaming on the gradient overlay network," in *Proc. of the IEEE International Symposium on Multimedia (ISM'10)*, pp. 1–8, 2010. Article (CrossRef Link)

[14] S. M. Y. Seyyedi, B. Akbari, "Hybrid cdn-p2p architectures for live video streaming: Comparative study of connected and unconnected meshes", in *Proc. of the International Symposium on Computer Networks and Distributed Systems (CNDS)*, pp. 175–180, 2011. Article (CrossRef Link)

[15] F. Wang, Y. Xiong, J. Liu, "mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast", *in Proceedings of the* 27 *International Conference on Distributed Computing Systems*, p. 49, 2007. Article (CrossRef Link)

[16] Q. Zhu, R.Wang, D. Qian, F. Xiao, "Re-exploring the potential of using tree structure in p2p live streaming networks," in *Proc. of the Sixth IFIP International Conference on Network and Parallel Computing (*NPC '09), pp. 125–132, 2009. Article (CrossRef Link)

[17] S. Asaduzzaman, Y. Qiao, 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, Jun. 2010. Article (CrossRef Link)

[18] S. Asaduzzaman, Y. Qiao, and G. Bochmann. "CliqueStream: An Efficient and Fault-Resilient Live Streaming Network on a Clustered Peer-to-Peer Overlay," in *Proc. of the 2008 Eighth International Conference on Peer-to-Peer Computing* (P2P '08). IEEE Computer Society, pp 269-278, 2008. Article (CrossRef Link)

[19] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, X. Zhang, "Inside the new coolstreaming: Principles,

measurements and performance implications," in *Proc of the 27th Conference Of the IEEE International Conference on Computer Communications (INFOCOM 2008)*, pp 1031-1039, 2008. Article (CrossRef Link)
[20] C. Hammami, I. Jemili, A. Gazdar, A. Belghith, M. Mosbah, "Hybrid Live P2P Streaming Protocol," in *Proc. of the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014),* Procedia Computer Science. vol. 32, Pages 158–165, 2014. Article (CrossRef Link)
[21] Omnet++ discrete event simulation system [online] (2012). URL http://www.omnetpp.org/
[22] Gt-itm topologies for the omnet++ simulation platform and oversim framework [online] (2010). URL http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html
[23] Star Wars IV Trace File [Online], 2012, url = {http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html}
[24] Y. Lu, B. Fallica, F. A. Kuipers, R. E. Kooij, P. V. Mieghem, "Assessing the quality of experience of sopcast," *International Journal of Internet Protocol Technology,* vol. 4, no. 1, pp. 11–23, 2009. Article (CrossRef Link)
[25] G. Gheorghe, R. Lo Cigno, A. Montresor, "Security and privacy issues in P2P streaming systems: A survey," *Peer-to-Peer Networking and Applications*, vol. 4, no. 2, pp 75-91, Jun. 2011. Article (CrossRef Link)
[26] H. Yifeng, A. K. Shujjat, "Improving streaming capacity in P2P live streaming systems via resource sharing," *Internet of Things and Cloud Computing*, vol. 1, no. 2, pp 15-22, 2013. Article (CrossRef Link)
[27] Z. Jianming, Y. Nianmin, C. Shaobin, L. Xiang, "Tree-Mesh Based P2P Streaming Data Distribution Scheme," *Knowledge Discovery and Data Mining Advances in Intelligent and Soft Computing*, vol. 135, pp 77-83, 2012. Article (CrossRef Link)
[28] NB. Ali, M. Molnar and A. Belghith, "Multi-constrained QoS Multicast Routing Optimization," *INRIA Research Report*, no. 6500, Apr. 2008. Article (CrossRef Link)
[29] W. Wu, J. Lui and R. Ma, "On Incentivizing Upload Capacity in P2P-VoD Systems: Design, Analysis and Evaluation," *Computer Networks*, vol. 57, no. 7, pp. 1674-1688, 2013. Article (CrossRef Link)
[30] L. Mengjuan, L. Fei, L. Xucheng, and Q. Zhiguang, "An ISP-Friendly Hierarchical Overlay for P2P Live Streaming", in *Proc. of 14-th IEEE International Conference on Peer-to-Peer Computing*, September 2014.

Chourouk Hammami is currently pursuing her PhD at the National School of Computer Sciences (ENSI), University of Manouba, Tunisia.
She obtained her bachelor degree in computer science applied to management from the High School of Management in Tunis (ISG) in 2007. She received her Master degree in computer science from ENSI in July 2010. Her research interests include computer networks, heterogeneous distributed applications and Peer to Peer (P2P) architectures and streaming.

Dr. Imen Jemili received her M.S. degree in Computer Science in 2002 from the National School of Computer Science (ENSI), University of Mannouba, Tunisia. In 2009, she received her PhD degree in computer science jointly from the University of Bordeaux 1 (France) and the University of Mannouba. Her research interests include wireless networks, ad hoc and sensor networks, power conservation, synchronization algorithms, routing, QoS management, simulation and performance evaluation. She is currently an Assistant Professor of Computer Science at the Faculty of sciences of Bizerte, University of Carthage, Tunisia.

Dr. Achraf Gazdar is currently an Assistant Professor at College of Computer Sciences and Information Systems at king Saud University. He received his PhD (2007) and MSc (2002) diplomas both in computer science from the National School of Computer Sciences (Ecole Nationale des Sciences de l'Informatique), University of Manouba in Tunisia and His Bachelor degree in computer science (2000) from the Higher School of Management (Institut Supérieur de Gestion), university of Tunis in Tunisia. His research focus on Video on Demand (VoD) systems design and architectures, video streaming systems, multimedia P2P networks and protocols.

Dr. Abdelfettah Belghith received his Master of Science and his PhD degrees in computer science from the University of California at Los Angeles (UCLA) respectively in 1982 and 1987. He is since 1992 a full Professor at the National School of Computer Sciences (ENSI), University of Manouba, Tunisia. He is currently on a sabbatical leave at King Saud University. His research interests include computer networks, wireless networks, multimedia Internet, mobile computing, distributed algorithms, simulation and performance evaluation. He runs several research projects in cooperation with other universities, research laboratories and research institutions. He is currently the chair of the IEEE Tunisia section, the chair of the IEEE ComSoc and VTS Tunisia Chapters, and the Director of the HANA Research Laboratory (www.hanalab.org) at the National School of Computer Sciences. He published more than 250 research papers in international journals and conference proceedings.

Dr. Mohamed Mosbah received his Ph.D. degree from the University of Bordeaux 1, France, in 1993. He was an associate professor between 1994 and 2002. He is a full professor in computer science since 2003 at Polytechnic Institute of Bordeaux, University of Bordeaux, France. His research interests include distributed algorithms and systems, formal models, security, and ad hoc and sensor networks. He participated to several national and European research projects, including collaborations with industry. He wrote more than 60 research papers published in international journals and conference proceedings and he is involved in various technical program committees and organisations of many international conferences.