Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 158 – 165

Procedia Computer Science

5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

Hybrid Live P2P Streaming Protocol

Chourouk Hammami[a], Imen Jemili[b], Achraf Gazdar[c,*], Abdelfettah Belghith[d], Mohamed Mosbah[e]

[a]*HANA Lab., University of Manouba, Manouba 2010, Tunis, Tunisia*
[b]*HANA Lab., University of Manouba, Manouba 2010, Tunis, Tunisia*
[c]*College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia*
[d]*HANA Lab., University of Manouba, Manouba 2010, Tunis, Tunisia*
[e]*LaBRI, University Bordeaux, Talence, France*

Abstract

A considerable part of the Internet traffic is due to Peer-to-Peer (P2P) protocols. The scalability of the P2P networks encourages implementing many P2P applications such as the file sharing, the media on demand and the live streaming. While many P2P solutions have been proposed the media freshness and the smooth media playback are still a challenging issues in the P2P live streaming. In this paper we propose the Hybrid Live P2P Streaming Protocol (HLPSP) which a live P2P streaming system based on a hybrid overlay (tree and mesh topology). The simulation results show that HLPSP outperforms the enhanced version of the famous CoolStreaming P2P system called DenaCast in terms of the startup delay, the end-to-end delay, the play-back delay and the data loss.

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

1. Introduction

Live video applications on the Internet have become more and more popular. These applications often require the collective use of massively distributed network resources and therefore are not adequately supported by the traditional client-server architecture in the Internet. 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 P2P live streaming systems include DONet/CoolStreaming[6], PPlive[1], PPStream[2] and many others as[10 12 12 4]. There are two widely adopted overlay architectures in P2P-based live streaming systems: the tree mechanism and the mesh mechanism. The tree-based protocols[18 3] build a tree overlay on the application layer. It borrows concepts from IP multicast. The media is pushed from the root to interior nodes to leave nodes. The short latency of data delivery is the main benefit of this approach. However, the tree structure is very fragile. Then the failure of nodes close to the root affects all the traffic that is forwarded by the interior nodes. An alternative to tree structured overlays is the mesh structure[5 8 10], in

* Corresponding author.
E-mail addresses: chourouk˙ham@yahoo.fr (Chourouk Hammami)., imenj1@gmail.com (Imen Jemili)., agazdar@ksu.edu.sa (Achraf Gazdar)., abdelfettah.belghith@hotmail.com (Abdelfettah Belghith)., mosbah@labri.fr (Mohamed Mosbah).

doi:10.1016/j.procs.2014.05.410

which the nodes are connected in a mesh-network. They use the availability of the data to guide the data flow. So peers can receive content data from multiple peers and provide data to multiple peers. Thus the mesh structure is highly resilient to node failures, but it is subject to unpredictable latencies due to the frequent exchange of notifications.
As the tree-based approach has deterministic delivery path and can confine the transmission delay effectively. Our goal is to find new design for the tree-based approach to remedy its shortages and retain its advantages. Our approach called Hybrid Live P2P Streaming Protocol (HLPSP) is based on the grouping of peers according to their upload capacity (a mesh topology). All groups form a multicast tree (tree topology). Each group represents a level of upload capacity. We assume that the source possesses the most upload capacity and belongs to level 0 (the highest level). The multimedia contents are delivered from the source being a member of the highest level to the low levels. That allows moving the powerful peers close to the source. The HLPSP is then based on a hybrid topology trying to assure the tradeoff between the tree and the mesh topologies benefits.
The rest of this paper is organized as follows: In section 2, we give the related works in this area. Section 3 presents HLPSP more specifically the overlay construction and a communication between the different actors. The performances evaluation of HLPSP is given in section 4. Section 5 concludes the paper and briefly discusses our future research plan.

2. Related Works

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 tree protocol organizes peers into a tree rooted at the source server. In this structure, nodes without children are called leave nodes and 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[18] and ZigZag[3] are examples of this kind of protocols. This single-tree based overlay suffers two drawbacks:1) potentially poor resource usage and unfair contributions in that a leaf node cannot contribute 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 approch[14 4] was introduced to tackle single-trees problems. 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 in which 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[1], DONet/Coolstreaming[6 8] and Prime[5] are examples of mesh based systems.
DenaCast[15] with which we compare our HLPSP performances is a mesh-based protocol. It is an enhanced version of the famous CoolStreaming protocol. It changes the way the CoolStreaming constructs the overlay. Within DenaCast, the tracker keeps a list of active peers in the network. Each peer requests neighbour from it. The later returns some random peers based on the number mentioned in the request message. When a peer receives the neighbours candidates list, it starts getting neighbours by JOIN_REQ, JOIN_RSP and JOIN_ACK messages. Periodically, each peer send notification message to the tracker in order to announce number of remaining neighbour they can get. We note that DenaCast didn't differentiate between peers capacity, each peer has a fixed number of neighbours. In addition, the neighbours send by the tracker are chosen randomly which causes maintaining a neighbourhood relations ineffective. Two main raisons have influenced our choice of DenaCast as a protocol to compare with: 2) it is an enhanced version of the famous CoolStreaming[8 9] system and 2) unless DenaCast, the lack of some details in the papers describing the previously cited works makes simulating them very hard to do.
To improve performance, some hybrid systems combine tree and mesh structures to construct a data delivery overlay such as[7 11 13 9]. For instance, the key idea of Treebone[7] is to identify a set of stale nodes to construct a tree-based

backbone, called treebone, with most of the data being pushed over this backbone.
While most of the hybrid solutions select some especial nodes (based on some performance criterions) to construct the tree-based overlay and then these nodes communicate with their neighbours using a mesh topology, our HLPSP organises 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 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.

3. Hybrid Live P2P Streaming Protocol: HLPSP

HLPSP organises the nodes on levels based on their uploads' 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 of level y ($1 \leq y \leq x$) can be served by nodes belonging to the same or to a higher level i ($1 \leq i \leq y$) called active nodes. A node of a given level j can serve nodes of the same level or less k ($j \leq k \leq x$) called passive nodes. In the following subsections, we give more details about the HLPSP overlay construction and the communication protocols between the different actors of the system.

3.1. Overlay construction

Three actors are involved in HLPSP the media source, the tracker and the node (the peer). Initially, the source sends its upload capacity to the tracker to registering. Then, the later calculates the maximum number of its passive neighbours denoted $NPmax$ ($NPmax = \frac{UploadCapacity}{PieceSize}$). From this moment 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 neighbours ($NAcurrent$) of a peer is less than its effectives neighbours ($NAeffective$), an effective neighbour is an active neighbour who really supplies the concerned peer by the pieces. In the reception of Neighbour request from a peer, the tracker responds by a Neighbour Response which contains the list of its available active neighbours. Besides, every peer sends periodically to the tracker an update message which contains the list of its actual neighbours.

3.2. Neighbour request handling

Upon receiving the Neighbour request from a peer p_i, the tracker verifies if it receives this message for the first time. If it is the case then it performs calculating $NPmax$, $NAmax$ which the maximum number of the active neighbours of this new peer ($NAmax = \frac{DownloadCapacity}{PieceSize}$). Once these two variables are calculated, the tracker assigns p_i to an appropriate level. So, the tracker possesses all the necessary information to fill a list called $Liste_{NA}$ with the maximum of the available active neighbours to be sent as a response to p_i. But if p_i has already sent a neighbours request, then the tracker calculates the number of the active neighbours to be sent back to p_i. The peers belonging to $Liste_{NA}$ have to satisfy these two conditions: 1) all the peers belong to levels having identifiers equal or less to that of p_i; 2) every peer guarantees the fact of serving p_i. That allows minimizing the number of messages exchanged during the connection phase. Also it guarantees the effective pieces download. Algorithm 1 details the filling list function.
First of all the tracker computes the threshold th_1 of the size of the $Liste_{NA}$. th_1 is equal to $NAmax$ for the newcomers. However, it is equal to the deference between the $NAeffective$ and $NAcurrent$ for the old peers.
Secondly, the tracker will look for the active neighbours to send. So, it must parse the levels starting by level 0 until level of p_i. For each, it tries at first to insert the peers which have a $NPmax > NPcurrent$ where $NPcurrent$ is the number of passive neighbours being served currently. It means that the peers have free places (line 3 in algorithm 1). When the tracker finishes the parse of each level, it must verify the size of $Liste_{NA}$. If the size of $Liste_{NA}$ is lower than th_1 and p_i has at least one free place, then the tracker restarts the parse of the current level in order to add any active peer p_j to the list having at least one passive peer with level greater than the level of p_i. This passive peer will be disconnected from his active peer p_j and connected to p_i as a new active peer during the connection phase of p_i to the

active peer p_j (lines 5 and 6 in algorithm 1). The passive peer address will be enclosed within the tracker response. After parsing all the levels the tracker verifies: if the size of $Liste_{NA}$ is still lower than a second threshold denoted th_2 which is the the average number of effective neighbours in the level of p_i; if yes it performs re-running the parse of levels. But in this case, it starts in reverse from level of p_i until level 1. For every level, it verifies if there are peers having not used their active neighbours; if it is the case, then it picks one active neighbour for every peer to be added to $Liste_{NA}$ (lines 14 and 15 in algorithm 1). Recall, in this second run, the tracker must update th_2.

Algorithm 1 Filling the $Liste_{NA}$ upon receiving a neighbour request

Require: Receiving a neighbour request from peerp_i
Ensure: Fill the $Liste_{NA}$
1: $level \leftarrow 0$
2: **while** $level <= level(p_i)$ and $sizeof(Liste_{NA}) < th_1$ **do**
3: Add any peer of level *Level* with at least one free place to $Liste_{NA}$
4: **if** $sizeof(Liste_{NA} < NAmax(p_i)$ **then**
5: Add to the $Liste_{NA}$ any peer of level *level* having at
6: least one passive neighbour with a less level than the p_i level
7: break if $Liste_{NA} = NAmax(p_i)$
8: **end if**
9: $level \leftarrow level + 1$
10: **end while**
11: **if** $sizeof(Liste_{NA}) < NAmax(p_i)$ **then**
12: $level \leftarrow level(p_i)$
13: **while** $level > 0$ and $sizeof(Liste_{NA}) < th_2$ **do**
14: Add to the $Liste_{NA}$ one active neighbour not belonging
15: to the $NAeffective$ of a given peer of level *level*
16: break if $Liste_{NA} = NAmax(p_i)$
17: $level \leftarrow level - 1$
18: **end while**
19: **end if**

3.3. Tracker response handling

Once the peer p_i receives the tracker response it starts establishing its neighborhood relations. Three type of messages can be exchanged between peers: the join request, the join response and the join deny. The join request is send by p_i to all the active neighbours enclosed within the $Liste_{NA}$ send within the tracker response. Upon receiving a join request, an active peer checks if the join request contains the address of one of its passive peers: 1) If yes, the active peer sends a join deny to its passive peer and encloses the address of p_i within the deny message. The passive peer can then perform the connection phase to its new active neighbour peer p_i by sending a join request to it. Finally, the active peer sends a join response to p_i to accept being its new active neighbour. 2) If no, the active peer just sends a join response to p_i to accept being its new active neighbour. Besides, two messages are exchanged between peers to download the media pieces: 1)chunk request sent by a peer to an active neighbour to ask for a data piece and 2) chunk response which contains the requested piece as a response to the previous request message. Note that we reuse the same piece and peer selection algorithms as DenaCast.

3.4. Update message sending

Based on its overall number of requested pieces, every peer classifies periodically its actives neighbors in effective and not effective neighbours. An updating message is then sent to the tracker containing the list of the non-effective neighbours to allow the tracker updating the overlay structure.

Periodically (two seconds), each peer sends to its passive neighbours a buffer map message containing the list of the media pieces available in its buffer. This message is also considered (by the passive peers) as a keep a live message to be ready for any departure incident.

4. HLPSP Performances' Evaluation

Table 1: Simulation parameter's list.

Parameter	Value
Peer buffer	40 s
Buffermap exchange period	1 s
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	200 s
Number of passive neighbours	Min (threshold(10), capacity upload/piece size)
Number of active neighbors	Min (threshold(10), capacity download/piece size)
Number of levels	6
Number of runs	10
Source number	1
Capacity of peers	10% (down 24 mbits0, 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)

4.1. Simulation setup

We consider 4 criterions in order to evaluate the performances of HLPSP: 1)The Data Loss which is the percentage of video content that is loss over the original video; 2) the End-to-End delay which is defined as the time between creating a frame in the source and playing it by the destination peer; We consider the average of all the peers in the network; 3) The Playback delay which is the average delay between the playback time by peers and the streaming time on the server; 4) The Startup delay which is the delay between the connection time to the mesh and the receiving time of the first buffer map. As we have discussed previously, the performances of HLPSP are compared to those of the enhanced version of CoolStreaming named DenaCast [15]. We conducted a set of simulations using the Omnet++ [16] simulator where DenaCast is already implemented. Network infrastructure used in the experiments is generated by GT-ITM module [17]Table 1 summarizes the simulation parameters.

4.2. Simulation results

Figure 1 represents the data loss percentage as a function of the peers number.

We clearly observe the sensitivity of DenaCast to the ADSL technology. As our protocol is 30 times more fluid than DenaCast. In fact a peer in our protocol always requests the pieces from the peers that are closer to the source than it. Indeed it has the opportunity to find easily its needed, while the supplier of pieces in DenaCast are selected randomly.

To ascertain the data freshness given by our protocol, we compute the end to end delay and the play back delay provided by each protocol. Figure 2a and figure 2b portray respectively the End-to-End delay and the Playback delay

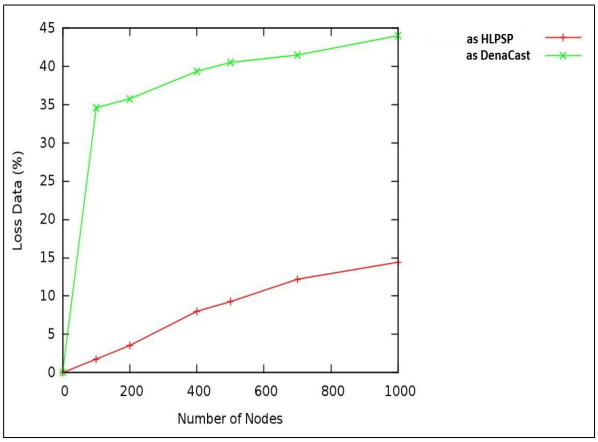
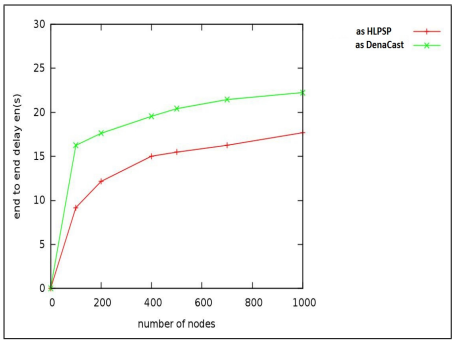
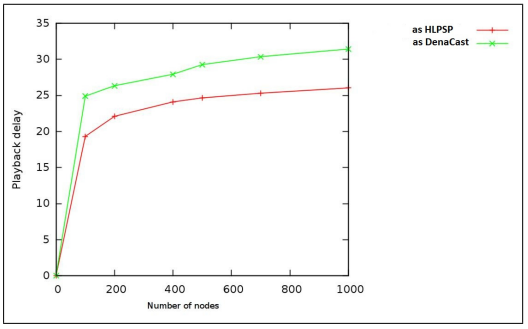


Fig. 1: The percentage of data loss as a function of peers' number



(a) The End-to-End delay as a function of the nodes' number



(b) The Playback delay as a function of the nodes' number

Fig. 2: The End-to-End delay and the Playback delay depending on the node' number

as a function of the nodes' number. We readily observe that HLPSP is more rapid in term of data reception by the peer and consequently the data playback than DenaCast. Indeed the download speed of pieces in HLPSP is higher than DenaCast for many reasons: 1) the number of actives neighbor of a peer is proportional of its download capacity in addition these neighbours are more closer to the source than this peer; 2) each peer exploits all its upload capacity in uploading peers which are away of the source. While a peer in DenaCast requests and serves the pieces to a fixed number of peers.

When nodes connect the network, the startup delay in DenaCast is higher than our protocol portrayed in figure 3.

Indeed in the later the tracker selects smartly the actives neighbors of a new arriving peer. So it guarantees that these peers will accept automatically the join request of this newcomer. While DenaCast selects the neighbors randomly. Hence the start up delay in DenaCast is higher. We note that the startup delay decreases when the numbers of nodes is equal 700 in the tow protocols. Since, in this case the select of neighbors by the tracker is more flexible.

Figure 4a and figure 4b illustrate respectively the signaling amount of data exchanged between the peers and between the peers and the tracker. As expected, DenaCast transmits less signaling messages than HLPSP. Furthermore, in the connection period of peers, HLPSP requires more signaling messages to be transmitted to assign peers in their levels and build relationships with their neighbours. However, as illustrated in figure 4b HLPSP transmits less signaling data to the tracker than DenaCast.

To ascertain the efficiency of our protocol in a dynamic environment, for 1000 nodes, we recomputed the data loss when 40% of nodes leave the network in different simulation periods. These period are respectively : uniform(30,50),

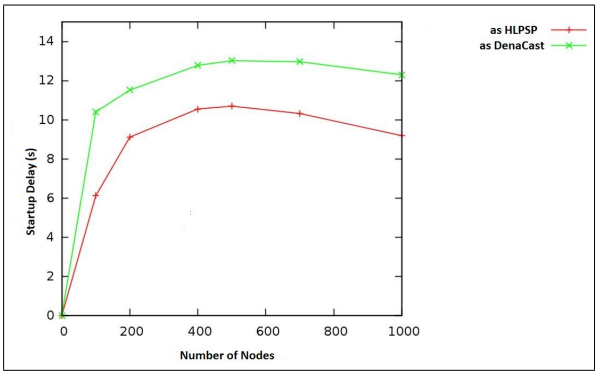
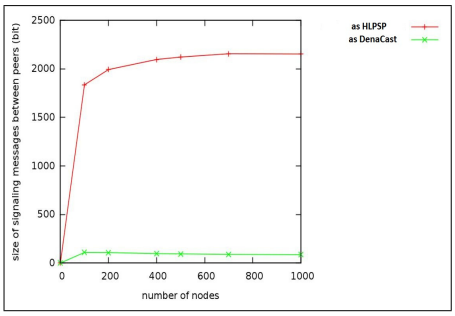
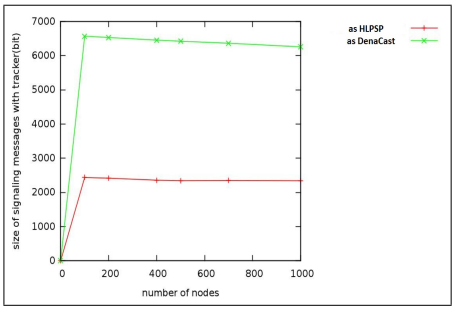


Fig. 3: The Startup delay as a function of the nodes' number



(a) Amount of data exchanged between the peers



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

Fig. 4: Amount of data exchanged within the HLPSP and the DenaCast networks

uniform(30,80), uniform(30,110), uniform(30,150), uniform(30,180). Figure 5 represents the percentage of data loss as a function of the leave time of these nodes. We clearly observe the sensitivity of the percentage of data loss to the leave time of nodes. As the nodes leave in advanced time of simulation, the percentage of data loss decreases. We have not considered DenaCast within this plot because of the lack of any mechanism to handle nodes' departure.

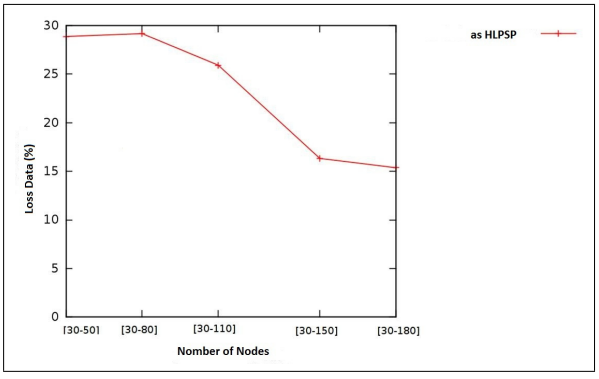


Fig. 5: the percentage of data loss as a function of the leave time of nodes

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 peer requests pieces from many parents according to its download capacity. Later it serves these pieces to many children corresponding to its upload capacity.

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

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.

References

1. Xiaojun Hei and Chao Liang and Jian Liang and Yong Liu and Ross, K.W., A Measurement Study of a Large-Scale P2P IPTV System,*Multimedia, IEEE Transactions on*,2007; p 1672-1687.
2. PPStream Team, 2013,www.ppstream.com/
3. Tran, D.A. and Hua, K.A. and Tai Do, ZIGZAG: an efficient peer-to-peer scheme for media streaming, *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 2003; p 1283-1292 vol.2.
4. Miguel Castro and Peter Druschel and Anne-marie Kermarrec and Animesh Nandi and Antony Rowstron and Atul Singh, Splitstream: High-bandwidth multicast in a cooperative environment,*In SOSP03*, 2003.
5. Magharei, N. and Rejaie, R., PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming,*INFOCOM 2007. 26th IEEE International Conference, 2007; p 1415-1423,*
6. *Xinyan Zhang and Jiangchuan Liu and Bo Li and Yum, T.P., CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming, INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE; 2005 p 2102-2111 vol. 3.*
7. *Feng Wang and Yongqiang Xiong and Jiangchuan Liu, mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast, Distributed Computing Systems ICDCS '07. 27th International Conference on; 2007, p 49-49.*
8. *Susu Xie and Bo Li and Keung, G.Y. and Xinyan Zhang, Coolstreaming: Design, Theory, and Practice, Multimedia, IEEE Transactions on, 2007; p 1661-1671 vol 9.*
9. *Bo Li and Susu Xie and Yang Qu and Keung, G.Y. and Chuang Lin and Jiangchuan Liu and Xinyan Zhang*, Inside the New Coolstreaming: Principles, Measurements and Performance Implications, *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008.
10. Purandare, Darshan and Guha, Ratan, An alliance based peering scheme for peer-to-peer live media streaming, *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*,2007.
11. Qinglin Zhu and Rui Wang and Depei Qian and Feng Xiao ,Re-exploring the Potential of Using Tree Structure in P2P Live Streaming Networks, *+Network and Parallel Computing. NPC'09. Sixth IFIP International Conference on*,2009; p 125-132.
12. Mol, J. J D and Bakker, A. and Pouwelse, J.A. and Epema, D. H J and Sips, H.J., The Design and Deployment of a BitTorrent Live Video Streaming Solution,*Multimedia, 2009. ISM '09. 11th IEEE International Symposium on*, 2009; p 342-349.
13. Asaduzzaman, Shah and Qiao, Ying and Bochmann, Gregorv, CliqueStream: Creating an efficient and resilient transport overlay for peer-to-peer live streaming using a clustered DHT, *Peer-to-Peer Networking and Applications*,2010; p 100-114.
14. ,Payberah, A.H. and Rahimian, F. and Haridi, S. and Dowling, J., Sepidar: Incentivized Market-Based P2P Live-Streaming on the Gradient Overlay Network, *Multimedia (ISM), 2010 IEEE International Symposium on*, 2010, p 1-8,
15. Seyyedi, S. M Y and Akbari, B., Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes,*Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*, 2011, p 175-180
16. Varga, A.,Using the OMNeT++ Discrete Event Simulation System in Education,*IEEE Trans. on Educ.*,November 1999
17. GT-ITM Topologies for the OMNeT++ Simulation Platform and OverSim Framework [Online],2010
18. , Banerjee, Suman and Bhattacharjee, Bobby and Kommareddy, Christopher, Scalable application layer multicast, *SIGCOMM Comput. Commun. Rev.*, 2002, p 205–217.