Hybrid CDN-P2P Architectures for Live Video Streaming: Comparative Study of Connected and Unconnected Meshes

S.M.Y. Seyyedi
Department of Computer Science and Eng., Sharif University of Technology, International Campus–Kish Island, Kish, Iran

B. Akbari
Department of Electrical and Computer Eng., Tarbiat Modares University, Tehran, Iran

Abstract— **There are two main scalable methods for streaming live video over the Internet: Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) networks. Though both have their own problems, P2P streaming systems challenge delivering video with constant quality and CDNs approaches require deployment of large number of servers throughout the Internet that is costly. Recently, using hybrid architectures based on both CDN and P2P networks has shown to be an efficient approach for large-scale video distribution over the Internet. This paper is compared the performance of two main hybrid CDN-P2P architectures includes: (i) *CDN-P2P unconnected mesh* in which independent P2P mesh networks are constructed under each CDN node, and (ii) *CDN-P2P connected mesh* in which CDN nodes and peers participate in construction of a single P2P mesh network. The comparison is preformed in addition, to the pure mesh-based P2P video streaming, using extensive simulation and based on different QoS metrics.**

Keywords-Peer-to-Peer systems; Hybrid CDN-P2P; Video streaming

I. INTRODUCTION

Media streaming applications over the Internet including user generated videos (e.g. YouTube) and IPTV are increasing every day. By increasing the users of media streaming applications, high performance, stable and cost effective infrastructures are necessary for large scale video streaming. Distribution of multimedia over the Internet in old fashion, client/server model, is very costly from the required infrastructures (servers and bandwidth) point of view.

Nowadays, there are two main approaches for large scale video streaming over the Internet: Content Distribution Networks and P2P architectures. In CDN architecture, video stream is delivered by IP multicasting or Application Level Multicasting (ALM) to large number of deployed servers on different part of the Internet. The users connect to a central server and redirected to one of the deployed CDN servers. Then, start streaming video from that server. Content provider companies use CDNs to distribute their media all over the world. Since the deployment of servers throughout the Internet by CDN providers is very costly, getting service from such providers cost high for content providers. Therefore, recently P2P streaming has gained more attention by researchers and also media content providers, but this architecture is a challenging approach for delivering video with constant quality and continuous streaming due to churn nature and limited resource of the Internet end systems. Therefore, it seems using a hybrid architecture based on CDN and P2P networks can be efficient approach for large-scale video distribution over the Internet.

Some recent researches (e.g. [1-2]) have dealt with integration of CDN and P2P approaches. In this assumption the whole CDN-P2P architecture could be considered as multiple *unconnected meshes* or one single *connected mesh*. This paper is compared these two approaches as well as traditional P2P mesh-based video streaming architecture based on Quality of Service (QoS) metrics in a large-scale network.

The rest of paper is organized as follows: reviewing related works in CDN-P2P area in section II. In section III, the paper describes briefly the three architectures including a pure mesh, hybrid CDN-P2P with unconnected meshes and hybrid CDN-P2P with connected meshes. Simulation setup and performance analysis are presented in section IV. Finally, the paper is concluded the paper in section V.

II. RELATED WORK

In this section the related work in CDN-P2P architecture for video streaming will be reviewed and tried to compare it with our work.

S. Banerjee et al. [3] propose a two-tier architecture for distributing media in large-scale networks like the Internet. They call this architecture OMNI. In this architecture there are two kinds of nodes, Multicast Service Nodes (MSNs) and end-hosts (or clients). MSNs could be considered as CDN servers. MSNs are distributed in the network to serve clients efficiently. Each end-host connects itself to one MSN and receives multicast data. To manage MSNs there is a distributed protocol. MSNs could form a multicast data delivery backbone on the Internet. Multicasting of MSNs nodes in OMNI could be application-layer multicast, network layer or series of unicast packets in client/server mode.

The paper claims that the proposed architecture is highly suitable particularly in real-time and delay sensitive (e.g. live video streaming) applications. In live streaming applications, unlike delivering data content, the media could not sent to servers (here MSNs) before the client starts to receive the

video chunks, so the appropriate method for this task is very sensitive. In live video streaming two things play an important role: a) Server load that is solved with MSNs in this architecture b) end-to-end jitter on client-side solved by organizing an overlay with minimum route's delay.

Since the goal of designing the architecture is minimizing end-to-end delay of clients, MSNs with more clients are more important than MSN with less. This importance could be changed by connecting or disconnecting clients. So OMNI should adaptively change the policy of distribution whenever the population of clients in one MSN changes. OMNI also uses a greedy algorithm to minimize the delay from source to MSNs.

Finally, it can be said that in OMNI an iterative solution is proposed to minimize the average delay in a distributed architecture. This solution is pure-decentralized because executing the greedy algorithm is operated only on the specified MSN. So there is no dependence between MSNs. This scheme enhances the quality of the tree architecture continuously based on the population of the MSN and leaves or joins clients. From architecture point of view OMNI uses tree approach for distribution in both MSN part and client part.

D. Xu et al. have proposed hybrid CDN-P2P architecture to limit use of the CDN servers in [4]. Therefore, this architecture is more cost-effective than pure CDN architectures. Both CDN and P2P have disadvantages when being used for media streaming. The P2P network is decentralized and video streaming is faced with jitter and it needs sufficient numbers of seeds to start playing video. On the other hand, in CDNs systems when the users increase the number of CDN server should increase too, that is costly. The proposed approach in this paper has advantages of P2P systems in cost as well as fast startup in CDN systems. It must be mentioned that the proposed architecture in this work is designed for video-on-demand not live streaming. It has also used mesh as an overlay between peers under CDN servers but there is no connection between these meshes.

The paper explains the scheme with one CDN; however, it is applicable to all CDN servers. In this architecture when a file needs to distribute, first it is delivered to a CDN, and the CDN allocates the resource and bandwidth to that file. When the file is requested from users, the server creates a virtual peer that is a seed. The requesting peers make partnership with the virtual peer and start to exchange the file. The virtual peer has high bandwidth and it could serve real peers efficiently. When the population of network grows, there are enough numbers of resources available in the network, thus the CDN server could leave the system and other P2P network could continue streaming without the CDN server. This action is called CDN-P2P *hand-off*. After the hand-off process the virtual peer is removed and the resource is released for alternative media. This paper has these contributions: Scheduling and planning for the capacity of CDNs and creating fairness between peers and preventing free-riding.

In [5] the authors conduct in depth analysis of CDN-P2P live video streaming based on the real deployed commercial system *ChinaCahe*. This system redirects clients to the closest server that located in users' service provider. This method is categorized in ISP-friendly and locality-aware approaches. To provide locality-awareness, they use DNS server that redirect clients based on their IP addresses.

In this system, the CDN servers are deployed in a multiple tree and organized in an overlay. In P2P part peer create a mesh under each CDN server.

This work considers join rates as to specify the time in which the user may arrive at the system. They also carry out an analysis on startup delay that the users experienced in the system. Measuring the system workload in different parts of it is done in this paper too.

In this paper the distributed video on the network is live. Also the mesh topology which this system uses supports locally-awareness but the meshes are independent of each other and there is no connection between them.

III. DESCRIPTION OF THE ARCHITECTURES

In previous section it was shown that all approaches use either tree or unconnected mesh for P2P part of network. Tree approaches suffer from resiliency problem. Therefore, the mesh approach is chosen for P2P part of the architecture. Our propose method is connected mesh that will explain in detail in rest of paper.

For comparison, three approaches for streaming live video are considered:

1. Mesh-based live video streaming that contains one source node and other peers that collaborate in the network with their upload bandwidths (like CoolStreaming [6]).
2. Multiple CDN servers that each acts as a source node. They serve mesh under themselves and like previous scenario peers in each network participate in distributing video in the network. In this scenario there is one separate mesh under each CDN server.
3. Multiple CDN servers and one tracker that create one big mesh in which all CDN servers and peer are taking apart on it.

All CDN severs work as a leaf in CDN Multicast network and as a source peer in the network. Since the architecture of CDN systems is optimized in commercial approaches therefore this paper is not working on structure of CDN system. Then ot tries to focus on the P2P part and transmission data from CDN server to P2P system. In every approach there are three different nodes: regular peers, source node (could be CDN servers) and tracker node. In following sections it tried to describe the architecture components and their roles.

A. Video coding

Coding of video is either variable bitrate (VBR) or constant bitrate (CBR). Since each frame is shown for constant amount of time on the screen, in VBR videos the size of frames is

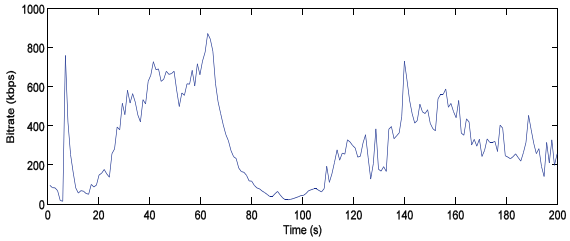

Figure 1. variable bitrates of a video trace file during time (Star Wars IV)

changing during the video. For example, when there are lots of changes in the scene, the bitrate increases. Since VBR coding is more efficient, this environment is used variable bitrate video. Fig. 1 shows the bitrates of Star Wars IV video tace file from [7] that is a variable bitrate. As it can be seen in this figure the bitrate of video is changing during time between 0 to 200 second.

In source node there is module called camera for encoding video. It stores the produced frames in its buffer for other peers' requests. In regular peers there is a module called player for reading video from peer's buffer and decode the video to calculate frame loss. The responsibility of player is calculation of frame loss ratio and distortion based on losses. The frame loss could happen in four situations:

1. Duo to late arrival: requested frame is received correctly but the playback time of the frame is passed.
2. Duo to dependency loss: the frame receives correctly but it is not possible to decode the frame, because it is dependent to other frames.
3. Due to available: the frame is not available in neighbors so it not possible to request it.
4. Due to network error: the frame is requested but it drops due to the network error.

B. *Tracker*

The responsibility of tracker in this system is the construction of mesh as well as connecting meshes under CDN servers in the last scenario. For mesh construction, the tracker keeps a list of active peers in the network. Each peer requests neighbor from tracker and send notification message to it in order to announce number of remaining neighbor they can get. Tracker always returns peers that connect directly or indirectly to the source, the ones at least with one neighbor. The process of mesh construction could be summarizing as follows:

- The source registers itself in the tracker.
- Peers start to enter the network and request peers from tracker.
- Tracker returns address random peers equal to number of required peer mention in request Message.

Mesh-based video streaming has simple algorithm. First, source node encodes live video and keeps it in its buffer and exchanges produced buffermap with its neighbors. The neighbors request video frames based on received buffermaps. Finally, peers that receive video equal to start-up buffering start to encode video its buffer and play. (Move up)

Input:
ServerNum: number of CDN servers in the system
Peerlist_size: number of active peers in the network
populationLimit: number in which each mesh get before connection start
connectedMesh: parameter that in decate the mesh will be connected or not

connection satisfier:
If connectedMesh = true **then**
  If (populationLimit × *serverNum*) > *peerList_size* **then**
    return false
  else
    return true
else
  return false

Output:
Boolean that specify the moment of start connecting meshes

Figure 2. Pseudo code for connection satisfier

When a peer arrives to the network, and request some neighbors from the tracker, it selects one mesh (select the one with less population) and return some peers based on the numbers mentioned in the message. When a peer receives neighbor candidate list, it start to get neighbor by JOIN_REQ, JOIN_RSP and JOIN_ACK messages.

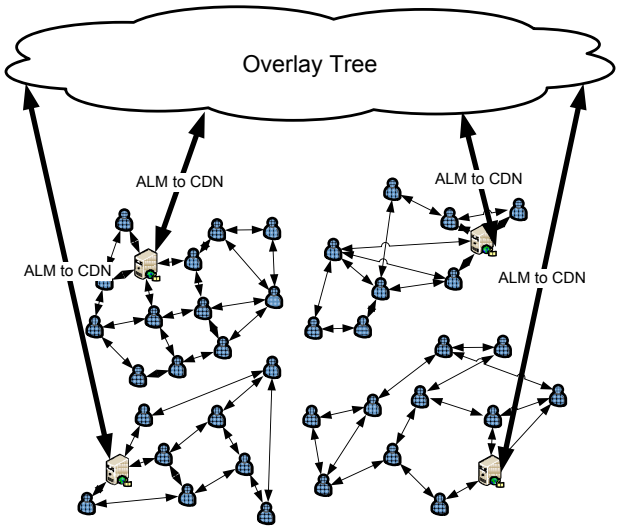

Figure 3. A snapshot of CDN-P2P unconnected mesh

The main role of the tracker is performing connection between meshes formed under each CDN. In the connected mesh scenario there is a parameter called *connected mesh* that if it becomes true, the tracker returns peers from other meshes after a time period. We call this period *connection satisfier*. This period is for preventing a peer to connect directly to two CDN servers at the same time. The algorithm to find this moment is shown in Fig. 2. After this period meshes become connected gradually and form a single mesh. A sample

snapshot of CDN-P2P unconnected mesh and connected mesh is depicted in Fig. 3 and Fig. 4.

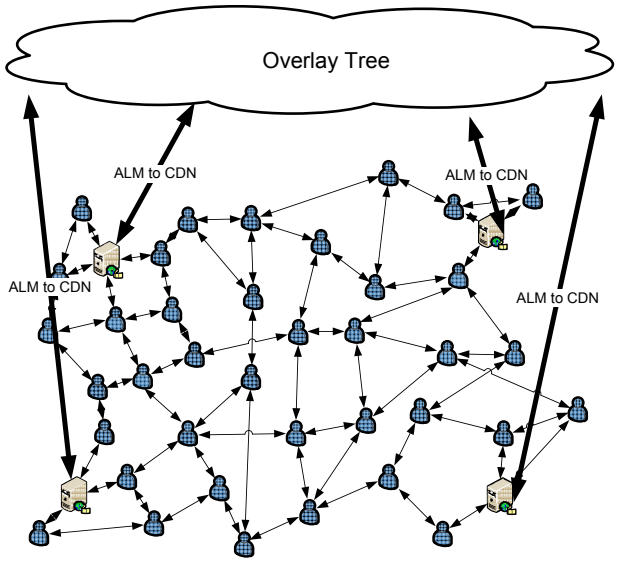

Figure 4. A Snapshot of CDN-P2P connected mesh

C. Scheduling

It could be said that scheduling is the heart of mesh-based P2P streaming. If there is no suitable scheduling, the protocol could not work perfectly. A scheduling works based on these questions:

- Which chunks is needed by the node's buffer to request?
- Which neighbors are able to supply these chunks for the node?
- How does the node choose the suppliers for requesting these chunks?

Since we are not working on scheduling in this project, a simple scheduling similar to what was proposed in [6] is used. Since the availability of frames in buffermap are exchanged, a frame is our chunk unit in this approach. Requesting one frame in the network brings huge overhead the scheduling, therefore, all requests to one supplier is sent in single message.

IV. Performance Analysis

For simulation the OMNeT++ v.4 [8] is used. OMNeT++ is a modular and discrete event simulator that is used for simulating communication network. The INET framework [9] is used for simulating TCP/IP network. INET framework implements UDP, IP and Data Link Layer in OMNeT++. OverSim [10] is a framework in OMNeT++ for simulating P2P systems. OverSim Uses INET framework for simulating underlay layers. OverSim prepares a reusable framework that has two main parts: 1) Overlay layer: for creating neighbor relation and constructing mesh (tree or structured systems), 2) application layer. The peer architecture in our simulation could be seen in Fig. 5.

In our simulation physical topology is generated using Georgia Tech Internet Topology Model (GT-ITM) [11] tools for OMNeT++ v.4 with 28 AS (backbone router) and 28 access router per AS in top-down mode and all use *drop-tail* queue. Peers select a router randomly and connect them by selecting random physical link with bandwidth between 1 Mbps to 2 Mbps and delays between 15 ms to 20 ms.

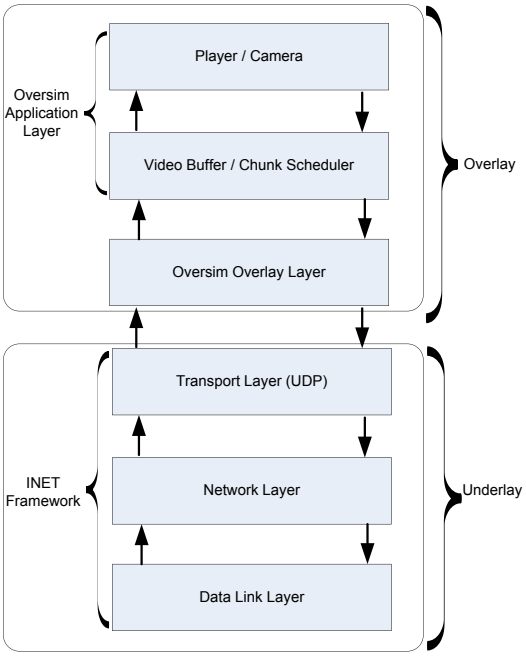

Figure 5. The peer architecture

For simulating valid video stream, the trace file from Video Trace Library in [7] is used. A chunk is a basic unit of partitioned streaming that is playable. Because of randomness property of P2P systems, all simulations are repeated 10 times and averaged all the peer's outputs for each scenario. The horizontal axis of all graphs shows the number of nodes participating in the network. Table 1 shows the rest of simulation parameters.

TABLE I. Simulation Parameters

Simulation parameters	Value
Maximum packet size	1000 Bytes
Peer side 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 video bit rate	512 Kbps
Number of neighbors	Random (3,5)
Simulation duration	200 s
Number of CDN servers in CDN-P2P	7

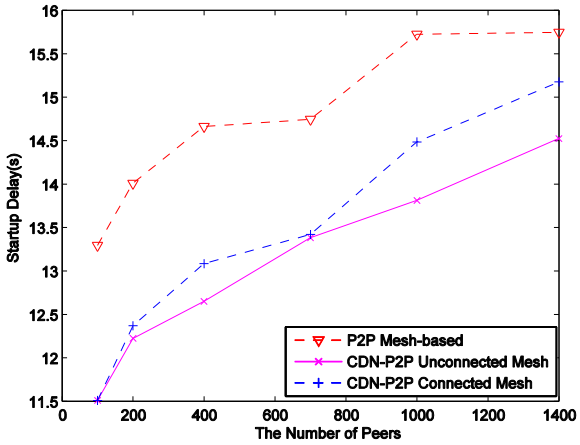

Figure 6. Startup delay against the number of peers

In this simulation to quantify the QoS of each scenario; some terminologies is defined and perform analysis of simulation results based on them. These metrics are startup delay, end-to-end delay, start playing time and distortion.

Startup buffering is the time of video that needed to be buffered before start to play. It is possible to set in configuration or estimate dynamically based on download rate. Delay between connecting to the mesh and start video playback is called startup delay. It is a random variable and depends directly on startup buffering and network delay. All the simulations with are done with 8 second startup buffering. Startup delay is evaluated in Fig. 6 As it can be seen CDN-P2P unconnected mesh has less startup delay in compare with other architectures.

Here end-to-end delay is defined as the time between creating a frame in the source node and playing it in destination node. This metric is increased as the network grows. This metric is averaged for all frames that each peer has received.

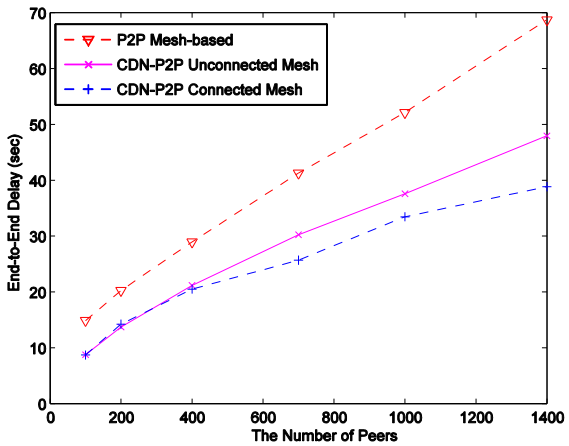

Figure 7. Average end-to-end delay in the network

In Fig. 7 it can be seen that end-to-end delay become lower for CDN-P2P connected mesh as network nodes increase. This is because nodes in a connected approach have more ways in average to reach CDN servers. Consequently the effect of having overlay routes with weak nodes could be more neutralized in such a network.

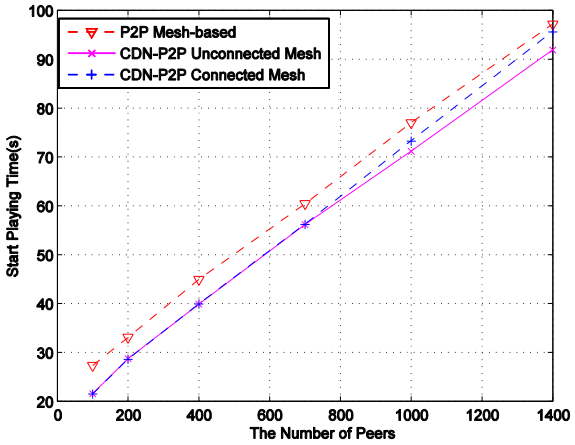

Figure 8. Start playing time against the number of peers

Start playing time is the time stamp in which a peer starts to play. This parameter increase as network grows. By attention in Fig. 8 it can be seen that CDN-P2P unconnected mesh has better performance in large network than other architecture for start playing time. It can also comprehend from this figure that by increasing the number of peers in the network the average start playing time in CDN-P2P connected mesh will be closely to the P2P mesh based approach.

Distortion is percentage of video content that is loss over original video. It could be obtained by the following formula:

$$Distortion = Total\ error\ frame\ size / Total\ frame\ size \times 100 \quad (1)$$

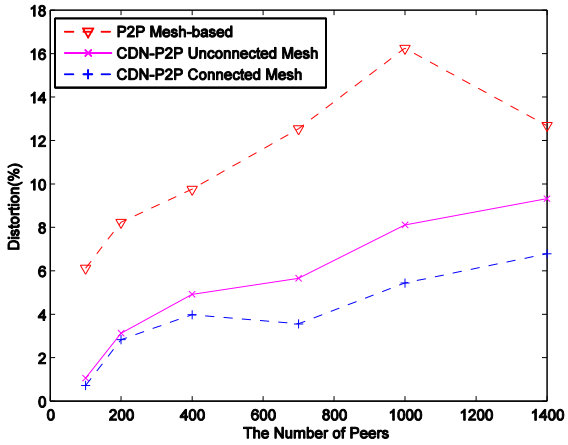

Figure 9. Average distortion of the received video in the group

Fig. 9 shows distortion of different architectures. Peer in connected mesh have better quality in average rather than other architectures. In all approaches the distortion increase as network grows. Although there are many ways for a node to retrieve the video but weak nodes in the network cannot supply resource well to their neighbors. This issue causes growth of distortion in larger networks. In connected mesh architecture

node have more chance to neighbor with strong nodes. In other words, the effect of weak nodes in this network is deducted. There is a reduction of distortion in P2P meshed-based graph after 1000 nodes. This is because of randomness of the peers (churn), and variable rate of video in simulation. Although the simulation is repeated 10 times and average the results, because of wide range of output, such a graph is produced for distortion. As an example the range of output values of distortion for 1000 nodes are range between 3.40 and 19.21 with different simulation seeds.

V. CONCLUSION

This paper compared the performance of CDN-P2P unconnected mesh and CDN-P2P connected mesh as well as mesh-based P2P video streaming through simulation. It had defined and illustrated the differences between these approaches. It was also explained about simulation setup and QoS for performance evaluation of these architectures. Finally, the simulation was performed and the graph of results is sketched. The result shows that CDN-P2P *connected mesh* architecture has significant enhancement in end-to-end delay and distortion. This architecture has higher startup delay in compare with *unconnected mesh* but this delay is tolerable for users.

REFERENCES

[1] D. Xu, S.S. Kulkarni, C. Rosenberg, and H.K. Chai, "Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution," Multimedia Systems, vol. 11, 2006, pp. 383–399.

[2] X. Liu, H. Yin, and C. Lin, "A Novel and High-Quality Measurement Study of Commercial CDN-P2P Live Streaming," WRI International Conference on Communications and Mobile Computing, 2009. CMC'09., 2009, pp. 325–329.

[3] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "OMNI: an efficient overlay multicast infrastructure for real-time applications," Computer Networks, vol. 50, 2006, pp. 826–841.

[4] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross, "A measurement study of a large-scale P2P IPTV system," Multimedia, IEEE Transactions on, vol. 9, 2007, pp. 1672–1687.

[5] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky," Proceedings of the seventeen ACM international conference on Multimedia, 2009, pp. 25–34.

[6] X. Zhang, J. Liu, B. Li, and T.S. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," proceedings of IEEE Infocom, 2005, pp. 13–17.

[7] "Video Trace Library," Sep. 2010.

[8] "OMNeT++ Network Simulation Framework" Sep. 2010.

[9] I. Vari, "INET Framework for OMNeT++," Sep. 2010

[10] I. Baumgart, B. Heep, and S. Krause, "OverSim: A scalable and flexible overlay framework for simulation and real network applications," Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on, 2009, pp. 87–88.

[11] "GT-ITM Topologies for the OMNeT++ Simulation Platform and OverSim Framework," Sep. 2010.