Peer to Peer Media Streaming Systems

Survey Paper

Yasmine Nader El-Glaly

VT, CS

yasmineg@vt.edu

Abstract

In recent years, audio/video streaming has become a popular class of applications and a major consumer of network bandwidth. In this paper, we survey the problem of streaming media systems through the internet using peer to peer overlay network. There are three main different topologies of peer to peer media streaming systems; the single tree, multiple tree, and the hybrid topology. We study three applications; each of them implement a different topology, so as we can see the advantages and disadvantages of each topology.

1. Introduction

1.1 Media Streaming Overview

In media streaming systems, a client consumes the content of a media file while the file is being downloaded, termed as the play-while-downloading mode [1]. At any time a client machine can request an audio/video file from a server. In most of the existing stored audio/video applications, after a delay of a few seconds the client begins to playback the audio file while it continues to receive the file from the server. The feature of playing back audio or video while the file is being received is called streaming. Many of the existing products also provide for user interactivity, e.g., pause/resume and temporal jumps to the future and past of the media file.

Streaming media enables real-time and continuous delivery of video and audio data in a fashion of "flow", i.e., once the sender begins to transmit, the receiver can start playback almost at the same time while it is receiving media data from the sender, instead of waiting for the entire media file to be ready in the local storage. Unlike normal data file, a streaming media file is huge, thus requires high channel bandwidth. Moreover, streaming media also carries stringent demand in the timing of packet delivery. The large size of the streaming media as well as its delivery timing requirement causes a streaming media server to be expensive to set up and run.

1.2 P2P Overview

P2P, as shown in figure 1, overlay networks are distributed systems in nature, without any hierarchical organization or centralized control. Peers form self-organizing overlay networks that are overlayed on the Internet Protocol (IP) networks, offering a mix of various features such as robust wide-area routing architecture, efficient search of data items, selection of nearby peers, redundant storage, permanence, hierarchical naming, trust and authentication, anonymity, massive scalability and fault tolerance. Peer-to-peer overlay systems go beyond services offered by client-server systems by having symmetry in roles where a client may also be a server. It allows access to its resources by other systems and supports resource sharing, which requires fault-tolerance, self-organization and massive scalability properties [2]. In P2P systems, there are no central servers. Every node acts both as a client and a server. This approach solves efficiently the scalability problem and distributes the load and the network bandwidth among all participating nodes or peers. This strategy solves also the bottleneck problem since no central server is responsible for handling all the incoming requests. Any peer in the system could respond to user queries given the necessary resources and computational capability.

A P2P system forms an overlay network where resources are shared among all participants. Information is also exchanged directly without the involvement of a third party and without the need of a centralized coordination. Another key feature of P2P systems is the volatility of the network connections. Peers operate outside the DNS, which is mainly characterized by its static nature, where nodes rarely change their topology. Peers can join and leave the P2P network at any time in a flexible manner without harming the functionality of other peers.

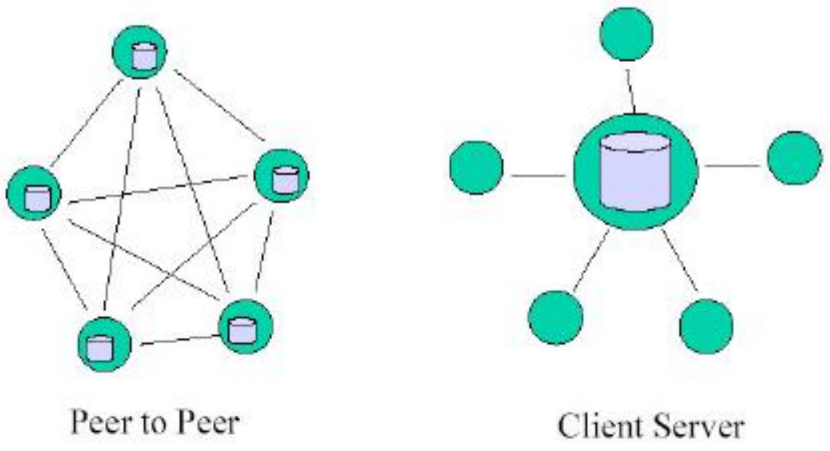

Figure 1: The Peer-to-Peer versus the Client/Server approach

1.3 P2P Streaming Media System

A simple and straightforward way of P2P streaming implementation is to use the technique of application-layer multicast (ALM). With ALM, all peer nodes are self-organized into a logical overlay tree over the existing IP network and the streaming data are distributed along the overlay tree. The cost of providing bandwidth is shared among the peer nodes, reducing the burden of the media server. In application-layer multicast, data packets are replicated and forwarded at end hosts, instead of at routers inside the network.

Building an efficient P2P streaming scheme, however, is truly a challenge due to several issues, including the following:

1) The end-to-end delay from the source to a receiver may be excessive because the content may have to go through a number of intermediate receivers. To shorten this delay (whereby, increasing the liveness of the media content), the tree height should be kept small and the join procedure should finish fast. The end-to-end delay may also be long due to an occurrence of bottleneck at a tree node. The worst bottleneck happens if the tree is a star rooted at the source. The bottleneck is most reduced if the tree is a chain; however, in this case, the leaf node experiences a long delay. Therefore, apart from enforcing the tree to be short, it is desirable to have the node degree bounded.
2) The behavior of receivers is unpredictable; they are free to join and leave the service at any time, thus abandoning their descendant peers. To prevent service interruption, a robust technique has to provide a quick and graceful recovery should a failure occur.
3) Receivers may have to store some local data structures and exchange state information with each other to maintain the connectivity and improve the efficiency of the P2P network. The control overhead at each receiver for fulfilling such purposes should be small to avoid excessive use of network resources and to overcome the resource limitation at each receiver. This is important to the scalability of a system with a large number of receivers [3].

In the rest of this paper, we will discuss the three topologies with a case study for each of them with the following order: End system multicast, Zebra, and Cool Streaming. Then, we show the differences among these applications; their strengths and weaknesses. And finally, open research issues in P2P media streaming applications are mentioned.

2. Comparison between P2P Media Streaming Systems

Several P2P streaming systems and algorithms have been proposed. P2P media streaming systems can be classified into three categories according to their architecture: central server based, distributed based, and hierarchy based. “Central server based” means that there is a central server in the system which is responses for the peer management and distribution tree construction. In comparison, “distributed based” system doesn’t have such central server, all the peer management operations and tree construction are distributed. The “hierarchy based” approach organizes peers into multiple layer hierarchical cluster, thus increases the system scalability.

According to the approach to organize peers and builds distribution tree, current systems can be classified into three categories, single tree, multiple trees, and mesh topology. Tree-based overlays have been a popular choice because a tree structure spans all peers, systematically avoiding the delivery of duplicate packets. In this approach, either one or multiple complementary spanning trees are constructed for data delivery.

2.1 Single Tree Topology

End System Multicast (ESM) belongs to this category. It is one of the pioneers of P2P media streaming systems [4]. ESM is a complete infrastructure for media broadcasting, implemented by Carnegie Mellon University. ESM is no longer under active development by researchers at CMU. In 2006, several members of the ESM research group founded Rinera Networks in order to commercialize the ESM technology. In 2008, Rinera Networks changed its name to Conviva. ESM allows broadcasting audio/video data to a large pool of users. The information is delivered following a traditional single-tree approach, which implies that any given peer receives streams from only one source. In IP Multicast data is delivered from the source to recipients using an IP Multicast tree composed of the shortest paths from each recipient to the source. Routers receive a single copy of the packet but forward it along multiple interfaces. At most one copy of a packet is sent over any physical link. Each recipient receives data with the same delay as though the end system were sending to it directly by unicast. Unfortunately, multicast is not available because of difficulties in implementation and agreements between ISPs. The only viable alternative is end-layer multicast. End System Multicast does not rely on router multicast. It abstracts the physical topology as a Complete Virtual Graph (CVG). Further it tries to construct a spanning tree of the CVG along which end system could send data to other recipients.It constructs an overlay tree to distribute data, and continuously optimizes this tree to minimize end-to-end latency. The root of the tree is the source of the broadcast. This is typically the machine that encodes the video data. This machine sends a stream of data packets to the nodes at the first level of the tree. Each of those nodes then forwards the data to the nodes connected to them, and so on, such that all nodes in the system receive the data stream. ESM is using "waypoints", machines from the PlanetLab test-bed [5], which help to increase the resource availability within the system and act as stable backbone for the overlay distribution tree. They also support a contributor-aware policy rather than a first-come-first-served approach. In a contributor-aware policy, the system knows which peers participate actively in the network based on resource availability or processing time. A contributor (peer which can get children) is then assigned more resources over a free-rider (peer who does not accept any child within the distribution tree or which is not so active). The first-come-first-served approach does not make any differences between peers. The first peer, which connects, gets the service. A key point is to automatically detect the capabilities of a peer in order to make best choices when downloading/uploading data from the network [6].

Their system is based on a single tree overlay which makes it highly sensitive to peer failures or disconnection. This project tries to handle receiver heterogeneity by prioritizing the audio streams, which is delivered at a low bit rate (20 kbps). A user with a 56K modem connection should then be able to receive at least the audio signal.

Conventional tree-based multicast is inherently not well matched to a cooperative environment. The reason is that in any multicast tree, the burden of duplicating and forwarding multicast traffic is carried by the small subset of the peers that are interior nodes in the tree. Most of the peers are leaf nodes and contribute no resources. This conflicts with the expectation that all peers should share the forwarding load. To address this problem, forest-based architecture is used as will be described in the next section, which constructs a forest of multicast trees that distributes the forwarding load subject to the bandwidth constraints of the participating nodes in a decentralized, scalable, efficient and self-organizing manner.

2.2 Multiple Trees Topology

A typical model of forest based P2P streaming system is Zebra [7]. Zebra is a streaming system implemented by the Massachusetts Institute of Technology (MIT). This application targets small and medium size networks (up to hundred nodes) and uses a two multicast tree streaming overlay. This application is simple and provides a good example of multiple trees based systems. Zebra system is divided into two parts: the server proxy and client proxies. As shown in Figure 2, the server proxy sits between the video server and client proxies. The client proxy sits between the client media players and the server proxy. Zebra's server and client proxies extend the Real Networks' application level RTSP proxy for UNIX. The key idea of this system is striping. It divides the constant stream of data into stripes to improve performance and robustness. In a peer-to-peer system, the stream of data is disrupted whenever a client leaves the system either due to a failure or a regular disconnects. Since clients receive pieces of the content from different senders, they can continue to receive some data even if one of the senders disconnects.

Data is sent using a peer-to-peer multicast scheme with striping. The video server sends one copy of the data to the server proxy, which forwards this data to two client proxies. These client proxies then distribute to other client proxies and their respective media players.

Each node (client proxy) is a source in one tree and a leaf in the other tree. A source forwards the received stream to one or more child peers. A leaf is located at the bottom of the tree, which implies that it has no children and for that reason does not forward any stream. When a disconnection occurs, only the direct children reconnect to the tree by contacting the server proxy. The server proxy is the main distribution source interfacing the streaming server. This server proxy has an important role in the Zebra system. It first manages the whole overlay.

The server proxy maintains the stripe distribution trees, lists of disconnected nodes, and some client state. In particular, it manages the following state for each stripe:

• The root node of the distribution tree.

• A list of disconnected nodes and their subtrees.

• The number of nodes currently serving in that stripe.

The server proxy manages the following state for each client node:

• The client's IP address and port number for messages.

• The stripe it serves.

• The additional number of nodes it can serve.

• A list of children nodes.

Zebra team states that their application supports at most hundred nodes, this small amount of users can be then handled centrally without much trouble. In addition, the server proxy performs media conversion on-demand. The proxy server is the main source of the multicast trees. It splits the data into two segments (called stripes) and sends them over two multicast trees. Finally, the proxy maintains and updates a complete list of child nodes. Zebra tries to reduce network traffic by grouping nodes close to each other. When a new peer joins the network, it first contacts the proxy server, which provides a random list of peers currently in the system. The peer sends then five ICMP messages to each peer and select the one with lowest average round trip time. Zebra reduces the required bandwidth for the server since it only needs to serve a single copy of the data. Zebra is successful in sending video data to all clients. It has been tested for up to 10 clients. For one test case, a server sent content at 40 kbps to 10 clients, showing Zebra allows a video source on a cable

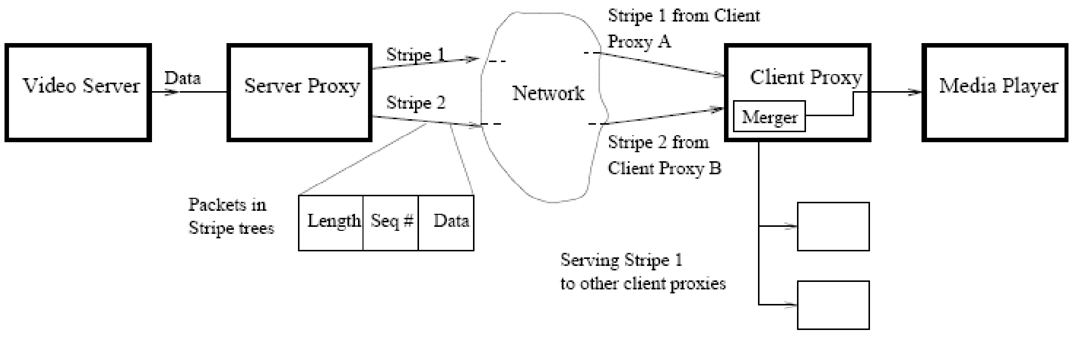

Figure 2: General system layout.

modem to broadcast video to 10 people. Furthermore, since it only requires command line configuration of the server address, Zebra is simple enough to be run by a regular Internet user. In order to evaluate how well the stripe technique increases tolerance to node failures, in a test case where a client completely loses one of its incoming stripes, and is unable to instantaneously reconnect. In this case, the client still received some data, allowing for a degraded level of service as opposed to no service. Video still appeared on the client media player, although it was choppy and included some artifacts. This test proves that data from the working stripe was able to be used by the client. Hence, striping proves to be an effective technique in improving system robustness.

2.3 Mesh Topology

In conventional tree-based P2P streaming architectures, at the same time peer can only receive data from a single upstream sender. Due to the dynamics and heterogeneity of network bandwidths, a single peer sender may not be able to contribute full streaming bandwidth to a peer receiver. This may cause serious performance problems for media decoding and rendering, since the received media frames in some end users may be incomplete. In forest-based systems, each peer can join many different multicast trees, and receive data from different upstream senders.

However, for a given stripe of a media stream, a peer can only receive the data of this stripe from a single sender, thus results in the same problem like the case of single tree. Multi-sender scheme is more efficient to overcome these problems. In this scheme, at the same time a peer can select and receive data from a different set of senders, each contributing a portion of the streaming bandwidth. In addition, different from the multi-tree systems, the sender set members may change dynamically, due to their unpredictable online/offline status changes, and the time-variable bandwidth and packet-loss rate of the Internet. Since the data flow has not a fixed pattern, every peer can send and also receive data from each other, thus the topology of data plane likes mesh. The main challenges of mesh topology are how to select the proper set of senders and how to cooperate and schedule the data sending of different senders. Examples of mesh-based multi-sender P2P streaming system include CollectCast [8], GnuStream [9], and DONet (CoolStreaming) [10].

CoolStreaming [10] is a data-driven overlay network for P2P live media streaming implemented by the Universities of Hong-Kong and Vancouver. This application coded in Python language creates its own overlay P2P network following a mesh topology. Figure 3 depicts the system diagram of a CoolStreaming node. There are three key modules: (1) membership manager, which helps the node maintain a partial view of other overlay nodes; (2) partnership manager, which establishes and maintains the partnership with other known nodes; (3) scheduler, which schedules the transmission of video data. For each segment of the video stream, a CoolStreaming node can be either a receiver or a supplier, or both, depending dynamically on the availability information of this segment, which is periodically exchanged between the node and its partners. An exception is the source node, which is always a supplier, and is referred to as the origin node. It could be a dedicated video server, or simply an overlay node that has a live video program to distribute. The key modules in the system are: l. Membership Management: Each CoolStreaming node has a unique identifier, such as its IP address, and maintains a membership cache (mCache) containing a partial list of the identifiers for the active nodes in the CoolStreaming. In a basic node joining algorithm, a newly joined node first contacts the origin node, which randomly selects a deputy node from its mCache and redirects the new node to the deputy. The new node can then obtain a list of partner candidates from the deputy, and contacts these candidates to establish its partners in the overlay. 2. Partnership Management: As said, neither the partnerships nor the data transmission directions are fixed in CoolStreaming. More explicitly, a video stream is divided into segments of a uniform length, and the availability of the segments in the buffer of a node can be represented by a Buffer Map (BM). Each node continuously exchange its BM with the partners, and then schedules which segment is to be fetched from which partner accordingly. 3. Scheduling Algorithm: Given the BMs of a node and its partners, a schedule is to be generated for fetching the expected segments from the partners. For a homogenous and static network, a simple round-robin scheduler may work well, but for a dynamic and heterogeneous network, a more intelligent scheduler is necessary. Specifically, the scheduling algorithm strikes to meet two constraints: the playback deadline for each segment, and the heterogeneous streaming bandwidth from the partners. If the first constraint cannot be satisfied, then the number of segments missing deadlines should be kept minimum, so as to maintain a continuous playback. CoolStreaming achieves a smooth video playback and a very good scalability as well as performance. The system has been extensively tested over the PlanetLab test-bed [5]. The overall streaming rate and playback continuity of CoolStreaming system is proportional to the amount of peers online at any given time. CoolStreaming does not follow some kind of distribution structure to deliver the media, but bases its delivery on data-availability.

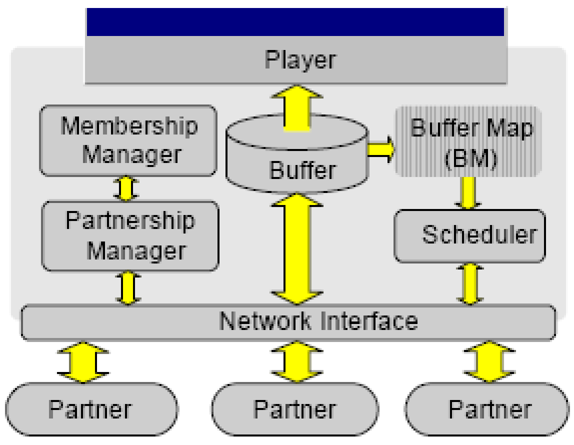

Fig. 3. A generic system diagram for a DONet node.

The application ran over all active machines of the PlanetLab test-bed (about 200 nodes). The average streaming rate was in the order of 500 kbps with a 60 segments sliding window (buffer map of 60 bits). Under dynamic environment (with many peer joins/departures), the overall control overhead was low: 1% of the whole traffic. During a live broadcast sport event, involving 50000 users, CoolStreaming system achieved a streaming rate between 450 kbps and 755 kbps using RealVideo and Windows media as formats.

3. Results

This survey does not present all available P2P streaming applications. Many other systems exist. The chosen applications describe different methods to perform P2P media streaming and are examples of different P2P streaming topologies. ESM and Zebra base their streaming overlay on a structural approach. Using trees, the flow of data is clearly specified from one node to the other. This approach, mostly adapted to live streaming, tries to reproduce the IP multicast scheme. However, these types of system suffer under highly dynamic networks and require complex algorithms to rebuild each tree so that the streaming session does not get interrupted.

Other solutions such as CoolStreaming do not follow any distribution structure and are based on data availability. Peers continuously notify each other when new data is available. The quality of the streaming session increases when the amount of peers in the overlay increases, since more peers mean more resources to choose from. Based on the achieved streaming rate, CoolStreaming and ESM are leading. Single-tree overlay suffers from many drawbacks such as sensitivity to peer disconnection and unfair data delivery method. ESM manages to overcome these problems and provides a well working application. Based on test results provided by each project, CoolStreaming achieves the best streaming rate and seems to be scalable since it has been utilized by a large amount of users (over 10000).

None of the listed software enables streaming to mobile devices. Their main focus is to deliver the best possible streaming quality to a large pool of users. The downside is that end-users need to have an advanced computer with broadband or Ethernet network access in order to process and view the stream properly.

A summary for the main differences among the discussed applications is illustrated in table 1.

Table 1. P2P media streaming applications comparison

	ESM	**Zebra**	**CoolStreaming**
Topology	single tree	multiple trees	mesh topology
Manufacturer	Carnegie Mellon University	Massachusetts Institute of Technology (MIT)	Universities of Hong-Kong and Vancouver
Number of supported nodes	Thousands	One hundred	Tens of Thousands
Streaming rate (Kbps)	100 to 300 for video, 20 for audio	40	500
Transport protocol	TCP	TCP	RTP
Split Stream?	No	Yes	Yes
Organization of peers	Central server-based	Central server-based	Distributed-based
Key features	1. Use NATs and firewalls as peers. 2. Prioritizing the audio streams.	1. Close peers are grouped together. 2. Users with limited bandwidth can stream live video.	1. The larger the data-driven overlay is, the better the streaming quality it delivers. 2. Less sensitive to peer failures 3. Distributing the data is simple, scalable and totally distributed.
Weakness points	Highly sensitive to peer failures or disconnection	Server proxy is a central point of failure	Membership management and scheduling algorithms are complex; cause overhead

4. Conclusion

Building an efficient P2P media streaming system confronts several challenges, including how to organize peers and build efficient distribution tree, how to handle peer failure, and how to adapt to the network dynamics. In this paper we investigate three popular live media streaming applications, End System Multicast, Zebra, and CoolStreaming. Each of them has its own overlay network topology. They use different algorithms in building the overlay tree of connected peers. According to the approaches to organize peers and build distribution tree, the P2P media streaming systems covered by this paper can be classified into three categories: single tree, multiple trees, and mesh topology. Single tree is the simplest architecture but it is very fragile. Multiple trees, though complicated in its design, it provides more robustness to the streaming system. Mesh topology has the advantages of both previous topologies as it does not have complex design and yet provides robust and scalable system. Scalability depends on the way the system organize the peers. CoolStreaming has higher scalability than ESM and Zebra because it is a distributed system; no central server is responsible for maintaining the states of the peers.

5. Open Research Issues

A lot of enhancements can be further introduced to P2P media streaming systems. We briefly are going to mention three issues that need further research work.

1. Security is one of the main recent research topics in P2P systems. Different possible attacks and vulnerabilities have been identified such as Attacks by self-replication, Man in the middle attack, denial of service attacks, routing attacks, Partition attacks. Research efforts have suggested different measures to increase the security of P2P systems that range from cryptography, to replication, to reputation protocols [11]. Still security remains a very challenging problem in P2P systems given the diverse and dynamic nature of these systems. Security models need to be intelligent enough to cope with the constantly changing environment of the P2P network.

2. QoS is another difficult issue that has to be addressed. In P2P streaming systems, a critical requirement is to operate the media distribution continuously. Hence, the difficulty resides not only in content location, but also in resource location, as peers need to discover which other connected hosts have enough throughput to act as forwarders and relay the media stream they have received. A Robust and Reliable P2P system should be able to support with an acceptable levels of QoS under following conditions: High churn, Node failure or departure, and Congestion in the interior of the network. Unavailability of stream content at play time causes jitter which degrades QoS. Also, end to end latency should be minimized. Scalability needs also to be improved to be able to accommodate a worldwide participation of users especially in the business world; increasing number of nodes should not degrade QoS.

3. As high-bandwidth wireless access becomes available everywhere through the wide deployment of wireless networks (WLAN, ad hoc, and 3G networks) and various wireless backhaul technologies (wireless mesh networks and WiMax), there will be a great demand on streaming applications such as news on demand through mobile devices. The techniques used in P2P media streaming could be applied in the wireless environment. However, unlike the Internet, connections in wireless

networks are even more dynamic and unstable. There are many challenges for wireless P2P systems such as limited radio bandwidth, limited battery, mobility, security, and so on. A common problem in wireless P2P systems is how to utilize P2P schemes in video streaming and schedule the video transmission among peers to minimize the "freeze-ups" in playback caused by buffer underflow in addition to the desire of energy efficiency. The Efforts are needed to cope with the challenges.

References

[1] D. Xu, M. Hefeeda, S. Hambrusch and B. Bhargava, "On Peer-to-Peer Media Streaming", IEEE ICDCS 2002, July2002.

[2] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma and Steven Lim, a Survey and Comparison of Peer-to-Peer Overlay Network Schemes, IEEE Communications Survey and Tutorial, March 2004.

[3]Duc A. Tran, Kien A. Hua, and Tai T. Do, A Peer-to-Peer Architecture for Media Streaming, IEEE Journal On Selected Areas In Communications, Vol. 22, No. 1, January 2004.

[4] Carnegie Mellon University. End System Multicast. http://esm.cs.cmu.edu.

[5] http://www.planet-lab.org.

[6] Yank hua Chu, Aditya Ganjam, T.S. Eugene Ng, and Sanjay G. Rao., "Early Experience With An Internet Broadcast System Based On Overlay Multicast", Technical Report, Carnegie Mellon University, 2004.

[7] Maya Dobuzhskaya, Rose Liu, Jim Roewe, and Nidhi Sharma, Zebra: Peer to Peer Multicast For Live Streaming Video. Technical Report, Massachusetts Institute of Technology, 2004.

[8] M. Heffeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," Proc. ACM Multimedia (MM'03), Berkeley, CA, Nov., 2003.

[9] X. Jiang, Y. Dong, D. Xu, and B. Bhargava. "GnuStream: A P2P media streaming prototype". Proc. the 2003IEEE International Conference on Multimedia and Expo (ICME'03), July 2003.

[10] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. "CoolStreaming/ Donet: A Data-Driven Overlay Network For Efficient Live Media Streaming". Proc. IEEE INFOCOM'05, 2005.

[11] Kaoutar El Maghraoui, " Survey of Peer-to-Peer Systems ", Technical Report, Rensselear Polytechnic Institute, Troy, NY, USA.