P2P Streaming Systems: A Survey and Experiments

Gustavo Marfia*, Giovanni Pau*, Paolo Di Rico†, Mario Gerla*
*Computer Science Department - University of California Los Angeles, CA 90095
†Dipartimento di Ingegneria Informatica, Elettronica e delle Telecomunicazioni, Universita‘ degli Studi di Pisa - Italy 56122
e-mail: {gmarfia|gpau|gerla}@cs.ucla.edu
e-mail: pdr1982@gmail.com

Abstract— **The great success of P2P systems for the purpose of file-sharing has set the path to likely be the next killer-app over the Internet, once the video streaming technology matures. Although solving scalability issues, P2P technology experiences problems of long start-time and churn-induced instability which can heavily affect the user experience. Moreover, technical and business solutions to Digital Rights Management are still under investigation. Great effort is being carried out in both academia and industry to solve these problems whose solution will offer a scalable, affordable and legal TV quality-like broadcast of content.**

In this paper we briefly analyze what is available to the end user in terms of P2P video-streaming products and try to understand which of these are the most promising for IPTV and content distribution companies.

In the following we produce (a) a survey of the available architectures, and; (b) a set of experiments on a popular peer-to-peer system.

I. Introduction

Overlays and P2P, initially developed to support IP multicast and file-sharing, have gone a long way beyond that functionality. These technologies have greatly enhanced the distribution of any type of information in the Internet by allowing efficient cooperation among end users. With the increasing bandwidth capacity provided by the Internet they are also proving to be key technologies for the delivery of real-time video and audio streams and of video-on-demand files.

Skype is one of the best examples of how P2P technology may be exploited in the consumer market. Beyond the Skype case, which relies on P2P technologies but seldom involves more than two users in a communication environment, new applications are emerging, that are more sophisticated in several aspects. Focusing on video streaming, we can distinguish two different scenarios.

One scenario corresponds to TV viewer migration from regular TV to IP-based TV to watch sport events, news, and popular broadcast channels. We may regard this process as a natural technological migration, where IP technologies add flexibility (e.g. choice between video-on-demand and video streaming) and interactivity to traditional TV service.

The other scenario corresponds to a growing community of end users that wish to share content often produced by themselves. An important contributor to this process is digital camera technology. A typical user uploads content related to his personal life (e.g. a birthday video) from his PC and distributes it within a community (e.g. friends). Such a customer is a *pro-sumer*, i.e., someone who produces and consumes content. Websites such as “youtube.com”, recently “Soapbox”, and “myspace.com”, are the main enablers of this phenomenon, they store and distribute content from/to millions of users.

With this background industry and academia are expecting a great success from the introduction and development of IPTV. Consumers will have the opportunity to use an interactive TV that either streams real-time content, or retrieves on-demand content. While an IPTV market barely exists today, industry analysts estimate there will be almost 80 million worldwide subscribers by 2011 [1]. Of the aforementioned systems, only Skype relies on P2P technology. The remaining systems unicast from server to receiver, an approach that does not scale well, unless they are backed up by sufficient server replication.

In this paper we explain how P2P solves scalability issues by exploiting its ability to distribute information. In the first part of this paper we survey available technologies for the support of video streaming. In the second part, we describe measurements and analysis based on an operational streaming system.

II. Background

There has been considerable work in the area of peer-to-peer live video streaming. We refer to only a small subset of the application layer work here omitting interesting approaches involving media encoding (e.g. [2]) and various network layer techniques (e.g. [3]).

P2P streaming systems strive to optimize three important metrics: SETUP delay (i.e. the time from when the user first tunes on the channel to when the video is visible), END TO END delay (i.e. delay between the content originator and the receiver, also known as playback delay) and PLAYBACK continuity (i.e. percentage of received data packets). Most of the systems may be classified based on the type of distribution graph they implement.

Reference Author: Gustavo Marfia, Computer Science Department, University of California Los Angeles, CA 90095, e-mail:gmarfia@cs.ucla.edu

This work is partially supported by the Italian Ministry for Research via the ICTP/E-Grid Initiative and the Interlink Initiative, the National Science Foundation through grants Swarms and Whynet, and the UC-Micro Grant MICRO 04-05 private sponsor STMicroelectronics. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

Tree-based overlays implement a tree distribution graph, rooted at the source of content. In principle, each node receives data from a parent node, which may be the source or a peer. If peers do not change too often, such systems require little overhead, since packets are forwarded from node to node without the need for extra messages. However, in high churn environments (i.e. fast turnover of peers in the tree), the tree must be continuously destroyed and rebuilt, a process that requires considerable control message overhead. As a side effect, nodes must buffer data for at least the time required to repair the tree, in order to avoid packet loss.

Mesh-based overlays implement a mesh distribution graph, where each node contacts a subset of peers to obtain a number of chunks. Every node needs to know which chunks are owned by its peers and explicitly "pulls" the chunks it needs. This type of scheme involves overhead, due in part to the exchange of buffer maps between nodes (i.e. nodes advertise the set of chunks they own) and in part to the "pull" process (i.e. each node sends a request in order to receive the chunks). Thanks to the fact that each node relies on multiple peers to retrieve content, mesh based systems offer good resilience to node failures. On the negative side they require large buffers to support the chunk pull (clearly, large buffers are needed to increase the chances of finding a chunk).

In the following we begin with a brief overview of popular tree-based systems and then focus on mesh-based ones.

A. Tree-based systems

Narada, by Chu et al. [4], is one of the first examples of end system multicast targeting video stream applications. In brief, Narada builds a mesh topology that connects the participating nodes selecting the links based on round-trip-time estimates between nodes. On top of the mesh, Narada builds a source rooted minimum delay tree used for media delivery. Narada has been implemented and tested with conferencing applications and is the underlying technology used in ESM (End System Multicast, `http://esm.cmu.edu`).

NICE, introduced by Banerjee et al. in [5], was initially designed for low-bandwidth, data streaming applications with a large number of receivers. Based on round-trip-time information between hosts, NICE builds a hierarchy of nodes. In this structure, nodes keep detailed knowledge of "close" peers (in terms of hierarchy) and coarse knowledge of nodes in other groups. No global topological information is needed. The hierarchy implies the routes.

Splitstream, by Castro et al. [6], improves fair sharing of resources among nodes. Tree-based systems, designed to limit end-to-end delay, tend to have a large number of leaf nodes. Leaf nodes do not contribute to the overall performance of the system. Splitstream fixes the problem by building multiple trees, where a node can be a leaf in all trees but one. Data, divided into stripes, is propagated using a different multicast tree per each stripe. A receiver, that wishes to attain a certain quality of service by receiving a certain number of stripes, joins the trees that correspond to those stripes.

B. Mesh-based systems

After the success of BitTorrent as a file-sharing P2P system, the same technology has been applied to streaming applications. At a high level the protocol works as follows. A new node registers to the system and receives the addresses of a set of trackers. A tracker is a super-node, which tracks which nodes are downloading or have downloaded a file. When the node contacts the peers advertised by the tracker, the node receives from each of them a buffer map, i.e., a map of the chunks of data they own and are able to share. At this point, based on various heuristics (e.g., bandwidth, delay), the node selects a subset of peers and requests chunks from them. This type of chunk exchange scheme is used in the below streaming applications.

PPLive is by far the most popular video streaming client. The protocol is proprietary, but thanks to recent work [7] it has been classified as a mesh-based scheme. The major difference with BitTorrent is that in PPLive packets must meet the playback deadline. In order to relax the time requirements, to have enough time to react to node failures and to smooth out the jitter, packets flow through two buffers: one is managed by PPLive, the second by the media player. A downside of such an architecture is the long startup delay. There are actually two types of delay: (a) the interval between channel selection and media display (10 to 15 seconds), and; (b) the playback time, required for fluent playback in spite of jitter (10 to 15 more seconds). The time lag between nodes may range up to about one minute. It could be distressing to hear nearby viewers scream "GOAL" while you are still watching the pre-goal action! Nevertheless, PPLive has proven to perform remarkably well in several important applications. On January 28, 2006 PPLive delivered a very popular TV program in China, hosting over 200K users, at data rates to users between 400 and 800Kbps, reaching an aggregate bit-rate of 100Gbps.

DONet (or Coolstreaming) is probably the next most successful P2P streaming system implementation to date. DONet, for features such as registration, peer discovery and chunk distribution, works the same as PPLive. Different from PPLive, however, DONet's creators published a lot of information about the internals of their scheme [8]. We may add, in addition to what has already been said for PPLive, that DONet implements an algorithm that chooses to download first the chunks with the least number of suppliers. In case of ties, DONet chooses the chunks owned by nodes with largest bandwidth.

Differently from above mentioned schemes, an Anysee [9] node participates in mesh building but it doesn't pull chunks from its peers. Every node in the mesh keeps an active path for data, and a set of backup paths, in case the active path fails to deliver within certain time constraints. Furthermore, this scheme first introduces the concept of inter-overlay optimization. Anysee involves all nodes (e.g. it uses the spare bandwidth capacity of the nodes that are receiving CNN to help those nodes that are receiving NBC) increasing network efficiency. Smaller buffers are then required compared to chunk-based schemes. In [9] the authors show experimental results based on the implemented system where the average delay between source and destination is within 30 seconds.

Below, we describe a brief set of experiments we performed on a popular streaming system, at least in terms of advertisement and channels.

III. Experiments

In both sets of experiments, our test host is equipped with an Intel Core 2 Duo Processor T5600 (1.83 GHz) with 2 GB DDR2 SDRAM. With such processor we are sure that any performance issues are not due to the host's capabilities, but to the network's capacity and to the p2p system's efficiency. In the following we will present what we observed during our experiments on a campus network and we will then briefly summarize the main differences with a residential network access.

A. Campus Experiments

At first we observe how packets are exchanged the very first time our client logs on the p2p network. This process involves an encrypted communication between the client and a server, where the client at first connects to a well known host to enter the network. This is typically performed in most p2p systems. Differently from other systems, where after a first login things may change, we here observe no changes in the pattern of exchanged packets. We then infer there is no attempt to delegate to peers successive logins and this process is always managed in a pure client/server fashion.

After the login phase, we are able to look at the data packets. UDP packets are received by our host during a video session. In each UDP data packet, the first two bytes represent an easily recognizable packet counter, while data, instead, is encrypted. By observing that data is not pulled from peers or servers but pushed to our client, we infer this may be a tree-based system. In order to be sure, we should be able to observe the whole network and derive its topology. We are anyway able to see that the most relevant peers involved in the streaming phase range from four to six. We should also emphasize that the most considerable part of the downloaded data, about 50 to 60%, mainly arrives from two subnetworks. This type of behavior suggests there is a two-tier hierarchy in the analyzed p2p system.

An interesting fact is that the system caches information, in a hidden folder, on the host's disk. The dimension of this file, about one gigabyte, is considerable. This is a different approach from what we typically observe in p2p streaming systems, it is evident that since content is on-demand, the system doesn't care about strongly caching and introducing high delays in the view of content. In order to understand the impact of this cache, we close and open the client a number of times and view the same content after each log on. At each step we observe a negligible download bit rate, about 80 kbps (which probably represents control traffic or missing frames) and an upload bit rate of about 136 kbps, almost all directed to one peer. We finally then switch to a new channel, a channel that has never been watched, so that it is less likely that it has been locally stored. In such case the upload portion of traffic substantially disappears and the download rate rises to about 640 kbps on average. Repeating such experiment in a number of trials, we observe that data is streamed at an average 640-670 kbps.

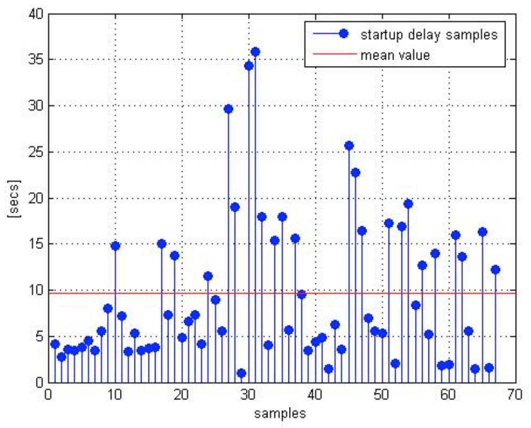

Fig. 1. Startup delay trials

Startup delay is measured as an average over fifty trials, testing a random channel on each trial. As we can see from Figure 1, this value is typically very low, about three seconds in most cases (i.e. we also observed a value of one second, evidently for a content partially saved on the client's hard disk), for an average of ten seconds and a maximum of thirty-five seconds.

In Figure 4 we can see that sometimes average data rate is higher than 640 kbps. This fact is due to upload rate. In fact plot refers to total rate. Typically we have a negligible upload rate. When we collected data plotted in this figure, cache was about 700 MB sized and we selected contents not stored in cache. Cache was not deleted (so we continued to have previous content encrypted in this file) and upload bit rate was 160 kbps on average. It means that some peers downloaded from my computer content stored on hard disk.

In the last test we try to observe if a scalable coding technique is used in the delivery of content. For this reason, we slowly lower the download capacity of our host. The only effect is the freeze of the video and the loss of frames. We can infer that no scalable coding is used, at least in this phase of the deployment of the streaming system.

B. Residential Experiments

In the residential setting we use the same client host, as mentioned, and access the Internet via an ADSL 2 Mbps connection.

We here summarize the main points in comparison to the campus settings as:

- Video quality heavily degrades, in particular the streaming rate abruptly varies from very low values, a few kbps, to about one Mbps;
- Startup delay does not change considerably;
- Upload rate is lower than in the campus settings case.

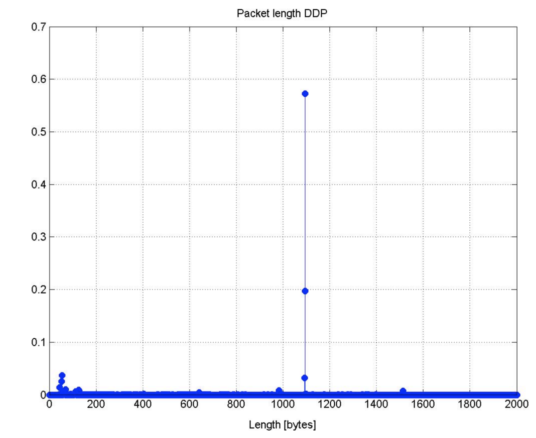

Fig. 2. Packet length DDP: about 75% are 1093-1094 bytes sized

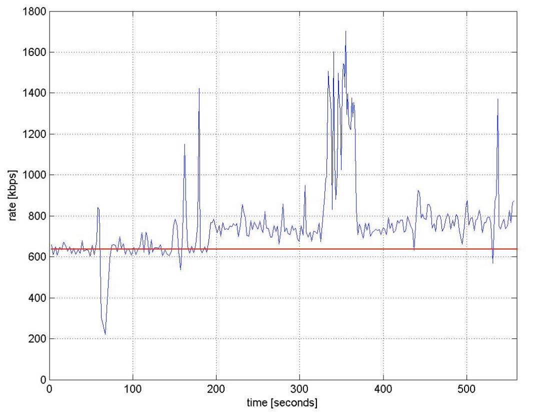

Fig. 3. Data rate: peaks correspond to channel switches; mean value is higher than 640 kbps (the red line) in this trace

IV. Conclusion

The aim of this work was to understand which are the most successful approaches to p2p content delivery and, with a few basic experiments, what is the behavior of a popular p2p streaming system. In the p2p system under test we find a very static and stable structure with a strong use of system's proprietary network. In addition, user's hosts are also relied on with a strong use of caching on the host's hard disk.

Although this is manly based on speculation, thus requiring a more extensive set of tests to better understand, we think the p2p system's topology structure is a hybrid of the CoopNet and the Anysee networks. Client-server traffic, as in CoopNet, is strongly used until the number of users is rather limited.

A number of more detailed experiments are necessary to better understand this p2p system. In particular, since this system will probably become the reference system on the Internet, due to its popularity, it is worth to investigate it with a more complete test campaign.

References

[1] [Online]. Available: http://www.cedmagazine.com/article/CA6412165.html

[2] X. Jin, K.-L. Cheng, and S.-H. Chan, "Sim: Scalable island multicast for peer-to-peer media streaming," in *IEEE International Conference on Multimedia Expo (ICME)*, July 2006.

[3] J. Li, "Peerstreaming: A practical receiver-driven peer-to-peer media streaming system," Microsoft Research, Microsoft Research TR-2004-101, Sept. 2004.

[4] Y. hua Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM Press, 2000.

[5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2002.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 2003, pp. 298–313.

[7] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," Department of Computer and Information Science at Polytechnic University, Tech. Rep., 2006.

[8] X. Zhang, J. Liu, B. Li, and T. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," in *IEEE INFOCOM,*, 2005.

[9] X. Liao, H. Jin, Y. Liu, L. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," in *IEEE INFOCOM*, 2006.