

A Survey on P2P Streaming Clients: Looking at the End-User

Alexandro Sentinelli(1,2), Luca Celetto(1), Damien Lefol (3), Claudio Palazzi (2),Giovanni Pau(2)

1) Advanced System
Technology,

STMicroelectronics,
Agrate Brianza (MI), Italy

Tel. +39 039 603 { 7600|7488 }

{alexandro.sentinelli |
luca.celetto}@st.com

2) Computer Science
Department,

University of California Los
Angeles (UCLA), CA, USA

Tel. +1 310 206-3212

{cpalazzi | gpau | alexsent}
@cs.ucla.edu

3) Livestation,

36-38 Hatton Garden

London EC1N 8EB

Tel.: +44 (0)20 7405 1444

{damien.lefol}
@livestation.com

ABSTRACT

Peer-to-peer (P2P) streaming systems grow in numbers and potential and several commercial products are already competing. Internet home users – through the diffusion of xDSL connections – represent the potential market of IPTV channels that Content Generators may distribute at reduced costs. This work describes the state of the art of P2P streaming clients and poses some questions about the end-user perspective which is still a non-trivial problem: expectations, content popularity, system's responsiveness and requirements. To this aim, a representative set of experiments has been performed on a popular p2p system. The client offers live streaming content from some European broadcasters, the start-up delay is just a few seconds and the user satisfaction rank is pretty good (resolution choice, good responsiveness, some popular channel). The new trend is to investigate flexible solutions in order to get closer to the user's needs and requirements. Unexpected cross-layer optimisations may overcome, like the synergic effect integrating video encoding techniques in a p2p environment. This work is aimed at helping the research community in getting a better comprehension of the issues and metrics that have to be considered in the design of p2p streaming applications.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Distributed applications.

Keywords

P2P streaming, Responsiveness, Scalable Video Coding (SVC), Heterogeneous Environment, End-User.

General Terms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WICON'08, November 17-19, 2008, Maui, Hawaii, USA.

Copyright ACM

1. INTRODUCTION

In TV Broadcasting the acronym of P2P (peer-to-peer) is often perceived like the panacea of cost balance sheets [1]. By exploiting the virtue of P2P scalability costs of infrastructure investments can be reduced in order to deliver contents to a huge set of users. There are, however, a lot of tradeoffs that need to be observed, though they surely bring considerable advantages. Investigation has evolved toward new approaches since the first Coolstreaming and relatives [3]. The scenario gets more complex when considering user expectations and needs. At this state of the art, we might say those mainly depend on the type of video content and the platform environment (network infrastructure, the rendering device). An important social event, for example, the soccer world cup, brings with it strict technological constraints such as the start-up delay, the video resolution and so on. Such constraints may become more flexible if the user is watching the news, weather, a music TV show, or an unpopular event. Moreover the streaming platform determines different user needs such as the resolution of the display, the cost of the network infrastructure (wired/wireless), or the computational power of the user device. In essence, user need/behavior has a huge impact on the protocol design. This led us to believe there cannot be a single optimal solution for all video application environments, or the most complex algorithm is also the most appropriate to fulfill the user expectations.

Nowadays several commercial products, like the one chosen for our case study, offer the same content at different qualities and bitrates to satisfy different sets of users. In general, the scenario can be very heterogeneous and involves a variety of fields and competences. For instance, an interesting synergy may overcome cross-layering Scalable Video Coding (SVC) in cooperative network environments [2]. SVC is an emerging video standard developed by the Join Video Team (JVT) to meet the various user need on heterogeneous platforms. This technique plays upon the concept of layered video coding splitting the main stream in many substreams, formed by a base layer with minimum quality (common to every peer) and a number of enhancement layers to increase the quality for more exigent users. The synergy produces better results than the state-of-the-art technology since this

solution allows distributing the same content to a larger portion of the overlay (the base layer represents a common content to the whole overlay).

The next section will describe the related work in the P2P streaming literature. It is followed by a brief set of experiments on a new successful client and a description of the qualitative synergy between SVC and P2P systems.

2. RELATED WORK

There has been considerable work in the area of P2P live video streaming. P2P streaming systems strive to optimize three important metrics: i) start-up delay (i.e. the time from when the user first tunes on the channel to when the video is visible), ii) end-to-end delay (i.e. the delay between the content originator and the receiver, also known as playback delay), and iii) playback continuity index (i.e. the counter of frames rendered in the right order by the player).

Most of the systems may be classified based on the type of distribution graph they implement. They can be classified in *tree* and *mesh* based; though a lot of hybrid solutions have been implemented already. Tree-based overlays implement a tree distribution graph, rooted at the source of the 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 frequently, such a system requires little overhead; in fact, packets can be 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 (nodes advertise the set of chunks they own) and in part to the pull process (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, as large buffers are needed to increase the chances of finding the missing chunks in the playback sequence.

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

2.1 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 follow. A new node registers to the system and receives the addresses of a set of trackers. A tracker is a super-node that tracks which nodes are downloading or have downloaded a file. When the considered 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 streaming applications described below.

PPLive is a very popular video streaming client. The protocol is proprietary, but thanks to a recent study [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 managed by PPLive and the second by the media player. Unfortunately, this architecture generates long start-up delays. More in details, two types of delay can be identified: i) the interval between channel selection and media display (10 to 15 s) and ii) the playback time, required for fluent playback in spite of the jitter (10 to 15s extra). The time lag between nodes may range up to about one minute. This is clearly unacceptable for popular events, if one considers how it could be distressing to hear nearby viewers connected with traditional systems screaming "GOAL" while you are still watching the pre-goal action! Nevertheless, PPLive has proven to perform remarkably well in several important occasions. For instance, on January 28, 2006, PPLive delivered a very popular TV program in China, hosting over 200 K users, at data rates between 400 and 800 Kbps, reaching an aggregate bit-rate of 100 Gbps.

DONet (or Coolstreaming) is another very successful P2P streaming system implementation [8]. This system works similarly to PPLive for features such as registration, peer discovery and chunk distribution. At the opposite from PPLive, its creators published a lot of information about the internals of their scheme. As a peculiar feature, 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 the largest bandwidth.

Differently from the aforementioned schemes, with Anysee nodes participate in building the mesh network but they do not pull chunks from other peers [9]. 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 introduces the concept of interoverlay optimization by involving all nodes in improving the global performance. For instance, it uses the spare bandwidth capacity of the nodes that are receiving CNN to help those nodes that are receiving NBC. Smaller buffers are then required compared to chunk-based schemes.

2.2 Tree-based

As one of the first examples of end system multicast targeting video stream applications, the system described in [4] proposes to build a mesh topology that connects the participating nodes by selecting the links based on round-trip-time (RTT) estimates between nodes. On top of this mesh, a source rooted minimum delay tree is built and used for media delivery. This solution has been implemented and tested with conferencing applications and is the underlying technology of real systems such as ESM (End System Multicast).

Nice [5] is another tree-based solution designed for low-bandwidth, data streaming applications with a large number of receivers. Based on RTT information exchanged among hosts, this solution builds a hierarchy of nodes; in this structure, nodes keep detailed knowledge of peers that are close in terms of hierarchy and coarse knowledge of nodes in other groups. No global topological information is needed.

2.3 Multiple trees scheme

In [8], it is shown how 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 generating unfair sharing of network resources among nodes. Splitsream fixes this problem by building multiple trees, where a node can be a leaf in all trees but one. Data, divided into stripes, are propagated using a different multicast tree for 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. Other advanced schemes such as CoopNet [10] and ChunkySpread [11] proposed to mitigate the strong dependency of a peer on all its ancestors in architectures based on a single tree. They are typically designed to work with more advanced video encoding techniques. For example, CoopNet uses Multiple Description Coding (MDC), which encodes a media stream into multiple independent descriptions. It constructs multiple independent multicast trees, one for each substream. A peer can improve its media quality by joining more multicast trees under the constraint of its download link capacity. More importantly, the departure of one ancestor in a multicast tree does not severely degrade the media quality of a peer, since it can still receive the majority of substreams from other multicast trees. These hybrid schemes (tree vs. mesh) tend to get the best features from the two approaches: robustness to high churn rate (mesh network) and a better efficiency (tree-based) in terms of traffic overhead through a more ordered distribution of requests.

3. EXPERIMENTS

For our case study we choose a live streaming client with good ranks from streaming-community forum and business-tech reviews. The client delivers video and audio content from both popular and unpopular European broadcasters. Some channels are available in 2 resolutions, to meet the user preferences and/or needs.

3.1 Setting

We used a HP laptop, Centrino processor, 512 DD RAM, Windows XP operating system with a commercial ADSL connection. What we measure in terms of statistics and performances is only related to the network traffic generated by the P2P client. Since the software does not allow our node to be the source of the video stream, observations have been performed only at client side. The main tools of this case study are Wireshark (Ethereal), a network analyser and Dumeter, a bandwidth monitor able to give a quick overview of the ongoing traffic (both upload and download). Other minor tools are a multitrack stopwatch, a bandwidth shaper, and PacketPlotter to export and analyze Ethereal traces on Windows Excel.

3.2 Chunk Size

Observing the traffic with Ethereal it is possible to extract useful information; we start with the vector size used to deliver the video stream. In Figure 1 we just counted the number of packets for each size for a short session with 1 Byte of granularity. In this graph two values are evident: 85 and 1397 B; the latter is the value of the packet size used to transmit the stream to the client.

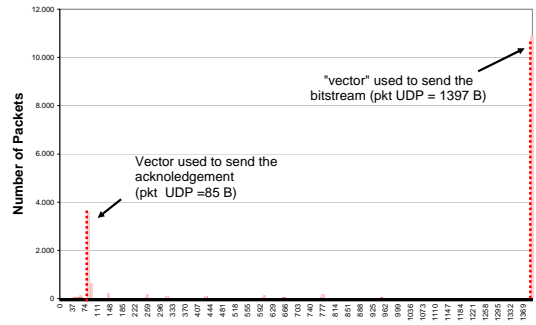


Figure 1: Short session: Pkt Length vs # pkt. 800 kbps.

The client uses, as usual for streaming application, the UDP protocol. Packets travel in this trace by groups of 3 up to more than 10. In long session traces there are evident sets of 3 packets at once every 85 Bytes packets. Similar patterns are easy to find in other P2P system. The smaller packet (85 here) is an acknowledgement for the chunk received which, for this client, should be around (or more) $3 \times 1397 = 4191$ B.

3.3 Start-up Delay

One of the biggest issues of P2P clients is indeed represented by the start-up delay. As a matter of fact, it represents the responsiveness of the system from a user perspective. With respect to other popular P2P systems, such as PPLive or Sopcast whose delay can be up to 2 minutes, this client never passes 10 seconds before getting a fluent stream.

3.4 Traffic Upload

Exporting with PacketPlotter Ethereal trace (Fig. 2) we get a shot of the download traffic per IP address for a short session.

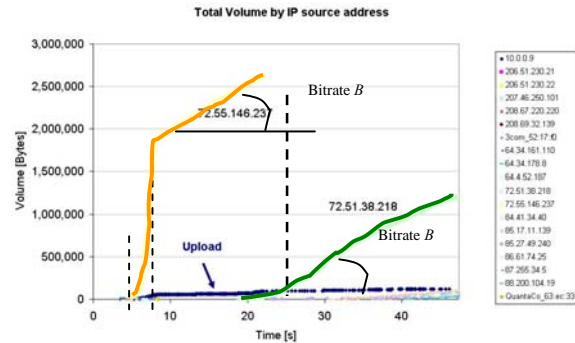


Figure 2; Dwnl traffic volume. 800kbps stream.

The download rate is higher at the start-up but then is always stable at rate B even when the node changes supplier (from IP address supplier 72.55.146.237 to 72.51.38.218). The traffic volume grows nicely linearly and the content streams fluent and smooth. Though, it is just remarkable that there's no upload for the majority of channels. It follows the same chart for the popular streaming client PPLive (Fig. 3).

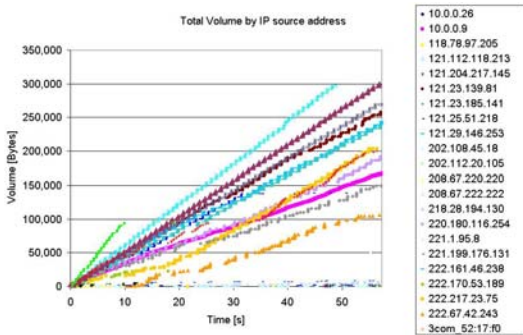


Figure 3 PPlive client - Dwnl traffic volume. 400kbps stream.

Solutions are both performing but designers faced eventually different constraints. PPLive is a pure P2P client where the infrastructure relies just on peers. The other client is a commercial product that has to deliver high quality live content at a remarkable bitrate (kbps 800 vs 400 for PPLive). At the moment there is no commercial Telecom provider able (or intending) to host a pure p2p network offering such a per-user bitrate. Skype worked by having the audio bitrate much lower than the video bitrate; this way, an infrastructure relying on common peers was still possible. In video streaming, either live or VoD (Video On Demand), the p2p approach is not negligible to reduce costs, but servers are still needed.

3.5 Surplus Bitrate at the start-up

If we focus our attention at the beginning of the session we clearly see a higher bitrate, approximately 1.5 Mbps.

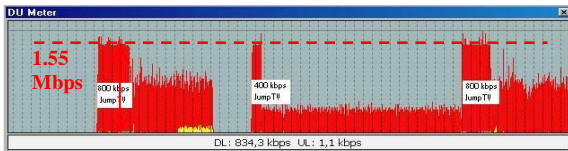


Figure 4: Start-up for the same channel at different bitrates.

We can measure the time interval I during which the high bitrate lasts before the step down to the bitrate of the bitstream and estimate the amount of data that the client collects to guarantee a smooth playback. From the user point of view the video starts after 5-10 s, but it keeps downloading at 1.6 Mbps for a time interval depending on the bitrate of the channel. This is possible only if the server delivers the stream with a playback delay bigger than the start-up delay perceived at client side; however, this also means that the stream is not pure live. In Table 1 we see the aforementioned values. We can play a bit with some easy math calculations to graphically represent the bitrate surplus at the beginning of each session. Physically, we have two flows to the buffer, one *in* that accumulates the stream (the download process), one *out* that empties the buffer. Then the stream is passed to the player and rendered on the display through a second buffer, the one of the video player (VLC). Although we need to isolate the effect of the p2p engine, a bit of caching is required by the system to work properly: 1 sec for the player's cache is a good trade off.

Table 1 Start session for different channels – Cache VLC 1sec.

Chan	Bitrate (kbps)	Start-up Delay (sec)	(*Interv. Higher bitrate (sec)	(**) Initial bitrate (kbps)
1a	400	2.3	15.00	1550.00
1b	800	7.0	40	1550.00
2	450	3.6	14	1550.00
3	400	7.0	15	1550.00

For these two flows we need two functions, respectively f_1 , the download bitrate, f_2 , the playback bitrate.

$$\begin{cases} f_1(t) = B_1 \text{rect}_{t_2} \left(t - \frac{t_2}{2} \right) + B_2 u_0(t - t_2) \\ f_2(t) = B_2 u_0(t - t_1) \end{cases}$$

We point out that after t_2 the two curves must give opposite and equal contributions to the size of the buffer. The following integral calculates exactly what is accumulated in the buffer from the start-up.

$$\int_0^{+\infty} (f_1(t) - f_2(t)) dt$$

If everything works fine the integral converges to an average value constant over time (otherwise the buffer is kept emptied or it adds up to infinite): this value corresponds to the surplus stream that the client has downloaded and is represented by the green *not* overlapped area (Figure 5).

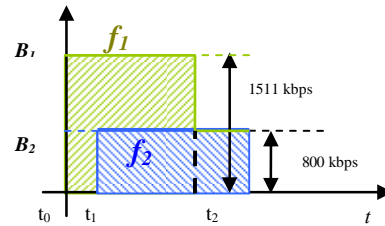


Figure 5: Traffic Surplus at start-up.

That is equal to:

$$B_1 \cdot (t_1 - t_0) + (B_1 - B_2) \cdot (t_2 - t_1)$$

where:

$$\begin{cases} t_0 : \text{Mouse click - channel selection} \\ t_1 : \text{Display rendering - Startup Delay} \\ t_2 : \text{The higher bitrate steps down to the stream bitrate} \end{cases}$$

With the same time legend, the integral of the above functions gives us the contribution to the buffer size over time for each flow; the middle red line in Figure 6 is what we are looking for. As any streaming client, the surplus covers a sort of guard interval that is big enough to smooth the bursty nature of Internet traffic and to guarantee an ordered sequence of data chunks (the problem appears when the node has more than one father).

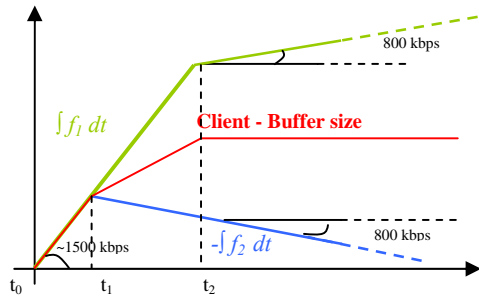


Figure 6: Client Buffer over time.

The solution here is as simple as efficient. The server stores at least $(t_2 - t_0)$ “live” content, which can be considered as relatively popular. If the user is not able to check the “reality” of the content the end-to-end delay loses importance. Instead, the start-up delay, i.e., the first angle of the red curve in t_1 , was moved back until a few seconds, giving the user a comfortable feeling of responsiveness. PPLive, Sopcast’s start-up delay can be up to 1 minute, unsustainable for a commercial application. This performance has been achieved through the use of servers carefully dimensioned to the overlay size. The P2P approach helps, but it represents just a contribution.

3.6 Heterogeneous environment

The heterogeneity of the network scenario determines as well different sets of users. The popular channels of our client are available at two resolutions offering the possibility for users to choose the quality they want. These streams are independently encoded, so the overlay is made by two independent sub-overlays, where each end-user belongs. Such solution does meet the user requirement and actually exploits already the virtue of P2P systems, however the selection of one fixed quality can be restrictive, for instance, in conditions with varying bandwidth availability (shared LAN, wireless,...). It is possible to improve this approach by adapting the quality stream on the fly. In fact, depending by the resources availability, the engine may switch from a low bitrate stream (or streamlet) to a higher bitrate as soon as enough low bitrate buffering has been performed to ensure continuous playback. If the higher bitrate cannot be maintained due to bandwidth limitations, the player switches back to a lower quality streamlet. This is possible only if several streamlets are being downloaded in parallel. Starting with the lower quality channel reduces the start-up delay (Cf. Table 1) and switching to higher quality once enough buffering is done can significantly improve user experience. Such an approach is used in [12]. The video is spliced before encoding and encoded at different bitrates. As splicing happens before encoding, each streamlet can be decoded independently. This enables the player to switch rapidly from one bitrate to another. The special encoding needed for this approach has the disadvantage of significantly increasing the end-to-end delay of the system. Moreover the regular switching of clients from one quality stream to another can make this technique impractical for p2p network as the high churn can affect efficiency.

The necessity to download several version of the same streamlet to ensure continuous playback is also wasteful of bandwidth. Scalable Video Coding (SVC) offers the best of both solutions by providing different qualities from only one stream. This brings

cross-layer optimisation between the video application and the network layer.

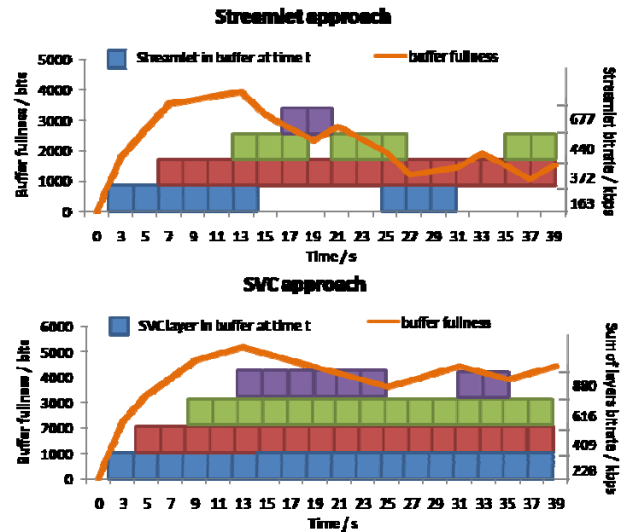


Figure 7: Switching from one quality to another using a streamlet approach (top) or SVC layers (bottom).

SVC is a layered encoding technique developed by the JVT committee to meet the video requirements in heterogeneous scenarios. As an extension compatible with the already existing AVC/H.264, SVC makes possible, for an Internet video provider, to generate and store a single version of the video, maintaining the ability to deliver HD to premium customers and scaled-down content to client with less capable connections. In the future, SVC will avoid the need to broadcast HD and SD version of transmission into separate channels, reducing the requests of bandwidth and freeing some frequencies for other application. This emerging standard is particularly suitable for IP networks. The Internet streaming scenario requires a coding standard that allows an easy adaptation to the frequent network fluctuations, a typical phenomenon due to the bursty nature of Internet traffic [13]. The H.264/SVC allows an adaptation that is as easy as dropping some of the information that is properly packed in Network Adaptation Layer Units (NALU), whose first bytes give the information about the scalability layer they belong to: practically speaking the main bitstream is cut and pasted. Even when the loss of compression efficiency due to scalability is taken into account, SVC improves user experience compared to streamlet. This is evident in Figure 7, where a SVC stream containing four layers is compared to four independent streamlets of similar quality. If comparing only the best quality streamlet to the SVC stream containing all layers, the bitrate of SVC is around 30% higher. However this loss of compression efficiency is more than offset by the gain of flexibility. No bandwidth is lost trying to download content, which cannot contribute to the improvement of quality. This means that the player is nearly always able to display the best or the second best quality layer, which is not the case with the streamlet approach. But there is another advantage we didn’t talk about. It is as interesting as much as unexpected the synergy encountered by the use of the SVC in P2P environments. Although the scalable video coding loses a bit in compression compared to a simulcast approach, the latter does not fully exploits the virtue of P2P systems. In fact, SVC encodes the high

quality stream as a base + enhancement layer. The two sub-overlays become an overlay embracing the entire peer population plus a smaller one delivering only the enhancement layer. In the previous solution the two classes of users couldn't share the base layer because the streams were independent (Figure 8, top). Through SVC (Figure 8, bottom) we get a common content shared in a much bigger overlay (the whole one). This means that, at least for the base layer, the research of good candidates is faster because every peer (so the closest ones) may share his own content and resources. The degree of cooperation increases and the load of requests is better distributed. This type of approach is also very well suited for commercial application based on heterogeneous p2p networks.

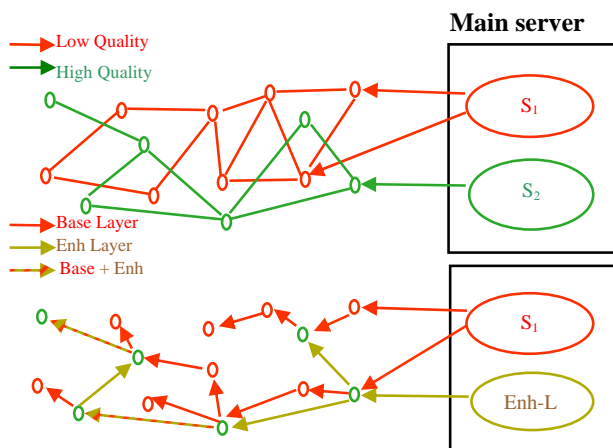


Figure 8: Overlay behavior in P2P networks using independent video encoding (top) vs. SVC (bottom).

These applications usually rely on a mix of servers or CDN backbone and p2p for distributing content. The backbone can then be used to insure that the base layer is delivered to all peers, and the peering is used to distribute enhancement layers. This enables a low start-up delay as the client connects directly to the server without waiting to find peers and the base layer stream is low bitrate. Once peers are located, the quality of the stream is improved by increasing the number of layers received. Relying only on the p2p network to distribute the base layer can be a risky strategy as without this layer not video can be decoded. Different techniques can be used to avoid using dedicated servers or CDN to distribute the base layer in a robust fashion. For instance, forward error correction (FEC) can be added or TCP can be used to distribute the base layer and UDP only for the enhancement layers.

4. Conclusion

The aim of this work is to understand the state of the art of P2P streaming design and particularly to describe new research trends in the area. The user experience in heterogeneous scenarios presents a new challenge. Designers approach the problem through cross-layer optimization and careful analysis of end-user needs and expectations. We have described the potential improvements of network efficiency using SVC at the application layer. Then our case study points out the importance of user experience and differentiation of system requirements. The user, depending on the type of content, may relax his expectations about the end-to-end delay but is still sensitive to responsiveness.

This is a non-trivial problem because different user expectations may change his behavior, which is a fundamental aspect to monitor in P2P environment. The optimal design incorporates a flexible implementation able to adapt to constraints efficiently and dynamically. There remains a lot of work to be done. The point of view of the user represents one of the key-drives for this new investigation approach where cross-layers and user satisfaction metrics are still to be further analyzed, optimized and, most likely, discovered.

5. ACKNOWLEDGMENTS

The work has been financed by the European Commission under the Seamless Content Delivery (SEA) project. We also acknowledge J. Hume from AST for his suggestions.

6. REFERENCES

- [1] http://www.eetimes.com/press_releases/prnewswire/showPressRelease.jhtml?articleID=X611872&CompanyId=1
- [2] T. Lee, Liu, Shyu, C.Wu, "Live Video Streaming using P2P and SVC", in MMNS, September 2008.
- [3] A. Sentinelli, G. Marfia, S. Tewari, M. Gerla, L. Kleinrock "Will IPTV ride the P2P stream?" in COMMUNICATION MAGAZINE, Jun 2007
- [4] Y. hua Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in SIGMETRICS '00.
- [5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in SIGCOMM '02:
- [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:
- [7] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," IEEE Transactions on Multimedia, December, 2007
- [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.
- [10] V.Padmanabhan,et.al."Supporting Heterogeneity & Congestion Control in P2P Multicast Streaming" in IPTPS 2004.
- [11] V. Venkataraman, K. Yoshida, P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast" in IEEE ICNP, 2006.
- [12] R.D. Major and M.B. Hurst, "Apparatus, System, and Method for Adaptive-Rate Shifting of Streaming Content", Patent US 2005/0262257 A1, 24th November 2005
- [13] H. Jiang , C. Dovrolis, "Why is the internet traffic bursty in short time scales?", 2005 ACM SIGMETRICS, Canada