A Survey on P2P Streaming Clients: Looking at the End-User (Invited Paper)

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

Internet home users – through the diffusion of xDSL connections – represent the potential market of IPTV channels that Content Generators may distribute at reduced costs thanks to Peer To Peer (P2P). 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 set of experiments has been performed on a successful P2P system. The new trend seems 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 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.

General Terms

Algorithms, Performance, Design, Experimentation.

Keywords

P2P streaming, Scalable Video Coding, Heterogeneous Platforms.

1. INTRODUCTION

In TV Broadcasting the acronym of P2P is often perceived like the panacea of cost balance sheets. Although P2P may solve (theoretically) the scalability issue a lot of tradeoffs need to be observed. 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 quality and so on. Such constraints may become more flexible if the user is watching the news or the weather. Moreover the streaming platform determines different user needs such as the resolution of the display, the cost of the network access (wired/wireless), or the computational power of the user device. In essence, user need/behavior has a huge impact on the protocol design. Nowadays several commercial products offer the same content at different qualities 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 P2P networks [1]. SVC is an emerging video standard developed to meet the various user need on heterogeneous platforms. This splits the main stream in a base layer with minimum quality and a number of enhancement layers for more exigent users. More adaptable to the available resources, SVC improves network cooperation since 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 follows a set of experiments on a new client and a description of the advantages brought by SVC in P2P streaming platforms.

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

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: *tree*, *mesh*, 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 bad 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

PPLive implements a mesh scheme, and is very popular video streaming client for movies, mangas, sports. 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. 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 (10 to 15s extra), which is unacceptable for popular soccer events (i.e. neighbours screaming “Goal” while you are still watching the pre-goal action!). DONet (or Coolstreaming) is another very successful P2P streaming system implementation [6]. 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 above schemes Anysee [7] 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

One of the first examples of end system multicast targeting video stream applications [2] 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 it, a source rooted minimum delay tree is built and used for delivery. Nice [3] 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 [4] it is shown how tree-based systems, designed to limit end-to-end delay, tend to have a large number of leaf nodes, which do not contribute to the overall performance of the system. Splitstream 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 schemes such as CoopNet [4] and ChunkySpread [5], in order to mitigate the strong dependency of a peer on all its ancestors in architectures based on a single tree, proposed cross-layer optimizations using 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 dwnl 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 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. In our experiment we used a HP laptop (Centrino processor), 512 DD RAM, Win XP operating system with an ADSL connection. The main tools of this case study are Ethereal, a bandwidth shaper and Dumeter, a bandwidth monitor able to give a quick overview of the ongoing traffic (upl and dwnl).

3.1 Network Traffic

In Figure 1 we get a shot of the dwnl traffic per IP address for a short session. The dwnl rate is higher at the start-up but then is always stable at rate *B* even when the node changes supplier.

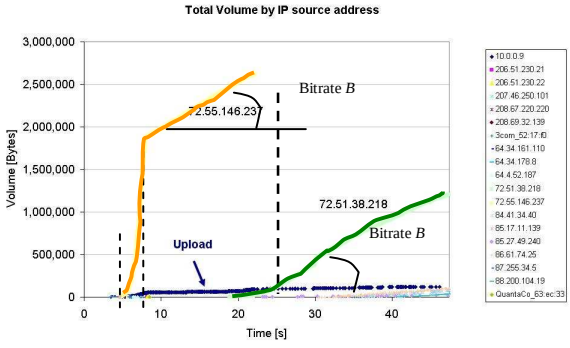

Figure 1 Dwnl traffic volume. 800kbps stream.

The traffic volume grows nicely linearly and the content streams fluent and smooth. Though, it is just remarkable that there's no upl for the majority of channels. In Figure 2, the same chart for the popular streaming client PPLive.

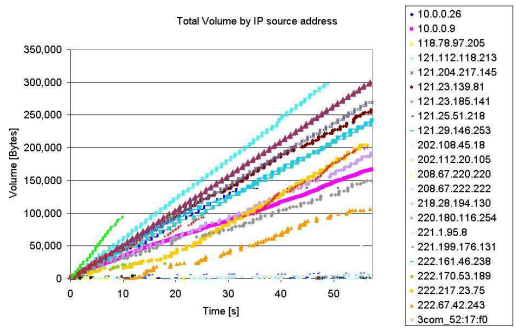

Figure 2 PPlive client - Dwnl traffic volume. 400kbps stream.

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 Telecom company able (or intending) to provide a sufficient upload bandwidth able to host a pure P2P streaming (either live or VoD) application. P2P helps, but servers are still needed.

3.2 Start-up

With respect to other P2P clients, whose delay can be up to 2 minutes, this platform never passes 10 seconds: a (still) comfortable time for the end-user.

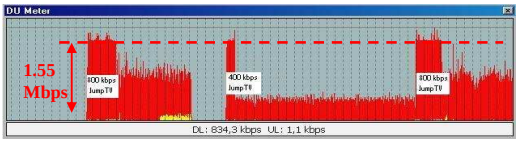

Figure 3 Start-up of 3 sessions.

In Figure 3 we observe a higher bitrate just after channel selection for a time interval *I*. We can measure such interval *I* before the step down to the bitrate of the stream. The video actually starts after ~5 sec, but it keeps downloading at 1.6 Mbps for a time interval depending on the bitrate of the channel. In Table 1 we see the aforementioned values.

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	1550.00
1b	800	7.0	40	1550.00
2	450	3.6	14	1550.00
3	400	7.0	15	1550.00

This is possible only if the server delivers the stream with an end-to -end delay bigger than the start-up delay perceived at client side (it also means that the stream can't be pure live). Physically, we have two flows to the buffer, one *in* (f_1) that accumulates the stream (the dwnl process), one *out* (f_2) that empties the buffer.

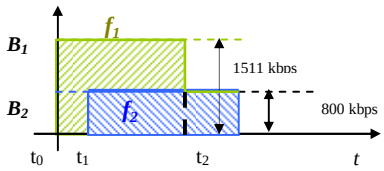

Figure 4 Traffic Surplus at start-up.

t_0 : Mouse click - channel selection
t_1 : Display rendering - Startup Delay
t_2 : The higher bitrate steps down to the stream bitrate

In Figure 4 the green *not* overlapped area corresponds to the surplus stream that the client has downloaded at the start-up. As any streaming client, the surplus covers a sort of guard interval to smooth the bursty nature of Internet traffic (or in case of temporarily network congestions) and to guarantee an ordered sequence of data chunks (especially when the node has more than one father). 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 looses importance. Instead, the start-up delay ($t_1 - t_0$) was moved back until a few seconds. 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. P2P, in this case, gives just a small contribution.

3.3 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 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, but we must ensure continuous playback by keeping the buffer not-empty. 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. This solution ensures continuous playback, but downloading several version of the same content is wasteful of bandwidth. Scalable Video Coding (SVC), instead, can provide different qualities from only one stream, and also brings an interesting optimization at the network layer. SVC is a layered encoding technique developed by the JVT committee to meet the 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 SD version content to client with less capable connections. This emerging standard is particularly suitable for IP networks where network fluctuations are frequent and unpredictable.

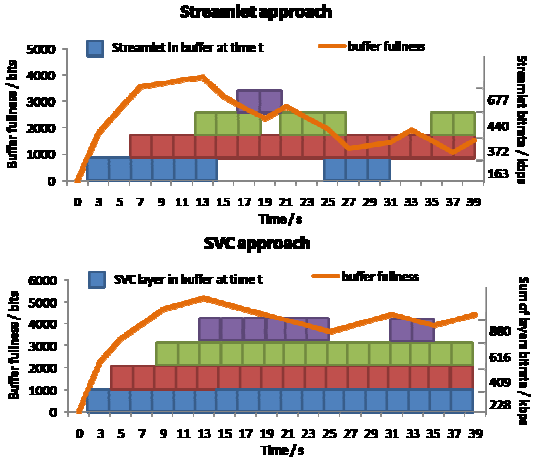

Figure 5: Switching from one quality to another using a streamlet approach (top) or SVC layers (bottom).

The H.264/SVC allows an adaptation that is as easy as dropping some of the information that is packed in Network Adaptation Layer Units (NALU), whose first bytes give the information about the scalability layer they belong to; in other words the down-scaled bitstream is extracted from the main one with a sort of "cut and paste" mechanism. 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 5, 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, but this is more than offset by the gain of flexibility and saving of bandwidth. Moreover, SVC brings an interesting and unexpected synergy if used 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.

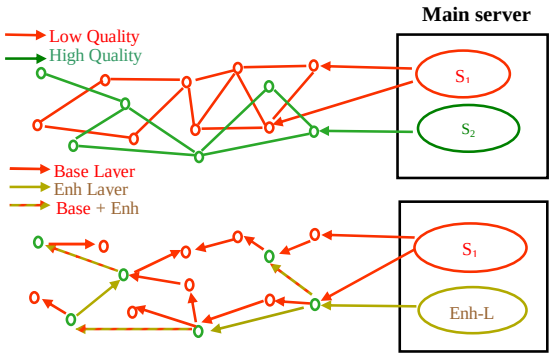

Figure 6 Overlay behavior in P2P networks using independent video encoding (top) vs. Scalable Video Coding (bottom).

If the broadcaster delivers two different qualities, in the previous solution the two classes of users cannot share the base layer because the streams are independent (Figure 6, top). Through SVC (Figure 6, bottom) we get a common content shared in a much bigger overlay: the two sub-overlays become an overlay embracing the whole one plus a smaller one delivering only the enhancement layer. This means that, at least for the base layer, the research of good candidates is faster because every peer can 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. 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 no 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 clients and particularly to describe new research trends in the area. Our case study points out that the user, depending on the type of content, may have different expectations about the end-to-end delay but is still sensitive to responsiveness. We also described the advantages of SVC in streaming platforms and its synergy with P2P. The point of view of the user represents one of the key-drives for this new investigation approach where cross-layers and user expectations/requirements 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 (AST's researcher) for her suggestions.

6. REFERENCES

[1] T. Lee, Liu, Shyu, C.Wu, "Live Video Streaming using P2P and SVC", in MMNS, September 2008.

[2] Y. hua Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in SIGMETRICS '00.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in SIGCOMM '02:

[4] V.Padmanabhan, H. Wang, and P.Chou, "Supporting Heterogeneity and Congestion Control in P2P Multicast Streaming" in IPTPS 2004.

[5] V. Venkataraman, K. Yoshida, P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast" in IEEE ICNP, 2006.

[6] 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

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