Survey on P2P Overlay Streaming Clients

Alexandro SENTINELLI[1(a, b)], Luca CELETTO[(a)], Damien LEFOL[(c)],
Claudio PALAZZI[(d)], Giovanni PAU[(e)], Theodore ZAHARIADIS[(f)], Ahola JARI[(g)]
[a] *Advanced System Technology, STMicroelectronics, Agrate Brianza (MI), Italy*
Tel. +39 039 603 {7600 | 7488} {alexandro.sentinelli | luca.celetto}@st.com
[b] *UCLA, CA, USA Tel. +1 310 206 3212, gpau@cs.ucla.edu*
[d] *Livestation, 36-38 Hatton Garden, London EC1N 8EB, UK*
Tel. +44 (0)20 7405 1444, damien.lefol@livestation.com
[e] *Università di Padova, Padova, Italy Tel. +39 049 827 1426, cpalazzi@math.unipd.it*
[f] *Synelixis Ltd., Chalkida, Greece, Tel+30 22210 61309, zahariad@synelixis.com*
[g] *VTT , Tampere, Finland Tel. +358 20 722 3334 Jari.Ahola@vtt.fi*

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 in heterogeneous networks. 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, start-up delay is a few seconds and the user satisfaction rank is good. The trend moves toward solutions that try to optimize the whole network stack, pursuing flexibility in terms of user needs and system requirements. This work is aimed at focusing on the key-drives in the design of P2P streaming clients.

Keywords. P2P Streaming, MDC, SVC, layered video coding.

Introduction

In TV Broadcasting the acronym of P2P (peer-to-peer) is often perceived like the panacea of cost balance sheets just by exploiting the virtue of P2P scalability. There are, however, a lot of tradeoffs that need to be observed, especially for high quality stream.

Investigation evolved toward new approaches since the first Coolstreaming and relatives [3] generating new products offered to consumers (e.g., Zattoo [1]). 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, (ex. 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 access (wired/wireless), the

[1] The work has been partially funded by the European Commission under the projects ICT-214063/SEA (www.ist-sea.eu) and the P2PNext (http://www.p2p-next.org/). Finally, the authors would also like to thank the Media Delivery Platforms (MDP) Cluster for reviewing the paper.

computational power of the user device. In essence, user needs have a huge impact on the protocol design. Nowadays several commercial products, like the one chosen for our case study, offer the same content at different qualities (thus, bitrate) to satisfy different sets of users. The scenario can be very heterogeneous and involves a variety of fields and competences. For instance, an interesting synergy may overcome cross-layered coding techniques (SVC/MDC) in P2P network that use multiple tree schemes. The synergy produces better results than the state-of-the-art technology since such solutions allows distributing the same content to a larger portion of the overlay (SVC) or makes the overlay more flexible to network congestions or channel zapping behaviour (MDC). In the next section we overview related work in the P2P streaming literature. Then we show a set of experiments on a new successful client. Section 3 describes the qualitative synergy between SVC and MDC in P2P systems. Finally, in Section 4, conclusions are drawn.

1. Related Work

The power of P2P is derived from several advantages in terms of robustness, reconfigurability and scalability.

From the broadcaster point of view, the P2P approach permits to serve a large audience without the need of additional resources. From the user point of view the P2P improves the visual experience by delivering video content and allows publishing own content with less (in some cases without) infrastructure costs and overcoming bandwidth/processing load bottlenecks. 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: mainly *tree* and *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 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.

1.1. Mesh-based Systems

BitTorrent's technology, after the success as a file-sharing P2P system, has been applied to streaming applications: now the client must meet the playback deadline. New nodes register and receive the addresses of the *trackers*, which track which nodes have downloaded a piece of content. When the node contacts the peers advertised by the tracker, the node receives 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.

PPLive is a proprietary popular mesh [7] video streaming client. 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 in spite of the jitter (10 to 15 s extra). The time lag between nodes may range up to about one minute, which is unacceptable for some popular events (i.e. neighbours screaming "GOAL" even just 2 seconds before you!). Nevertheless, PPLive has proven to perform remarkably, for instance, on January 28, 2006, PPLive delivered a popular event in China, hosting over 200 K users, at data rates between 400 and 800 Kbps.

SopCast builds a mesh too [3], but enables easily anyone with an ordinary broadband connection to broadcast their own contents. Start-up delay can be from 15 seconds up to 2 or 3 minutes: it strongly depends by the type of content streamed, because that determines the size of the overlay, thus the availability of upload resources.

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 inter-overlay 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.

1.2. Tree-based Systems

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.

1.3. Multiple Trees Systems

A tree-based system, designed to limit end-to-end delay, tends to have a large number of leaf nodes that do not contribute to the overall performance of the system generating unfair sharing of network resources among nodes. *Splitstream* [6] 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. Some of the above protocols represent also an interesting example of cross-layer optimization between the application and network layer. Layered video coding techniques, in fact, merge conceptually with the idea of splitting the main content in several sub-streams, fitting the variety of user needs in heterogeneous environments.

2. Case Study

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.

2.1. 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. Other minor tools are a stopwatch, a bandwidth shaper, PacketPlotter to export Ethereal traces on Windows Excel.

2.2. Network Traffic

Exporting with PacketPlotter Ethereal trace Figure 1 we get a shot of the download traffic per IP address for a short session.

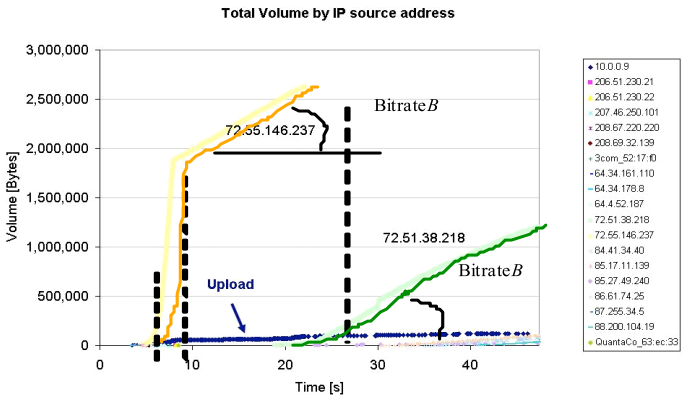

Figure 1. Download traffic volume. 800kbps stream.

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 (thus it works more like a Client-Server model). It follows the same chart for the popular streaming client PPLive (Figure 2). 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 (800 vs. 400 kbps 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. In video streaming, either live or VoD (Video On Demand), the P2P approach is not negligible to reduce costs, but servers are still needed.

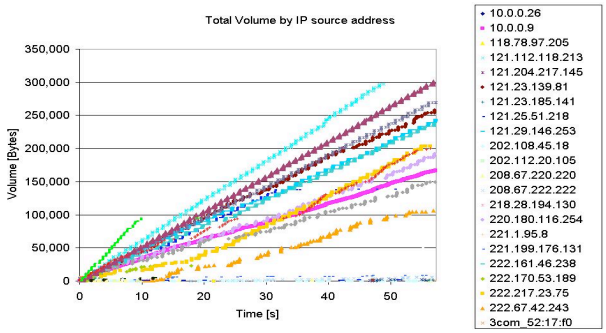

Figure 2. PPlive client - Download traffic volume; 400kbps stream.

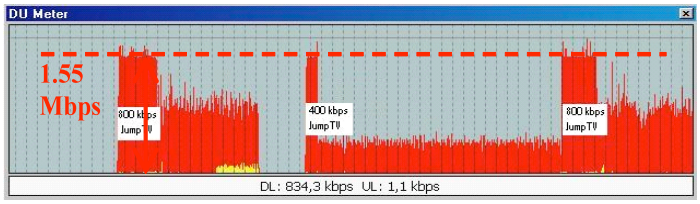

Figure 3. Start-up for the same channel at different bitrates.

2.3. Surplus Bitrate at the start-up

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, this client never passes 10 seconds before getting a fluent stream. We tried to understand what happens just monitoring the traffic at the beginning of the streaming session, thus we clearly remarked a higher bitrate, approximately 1.5 Mbps (Figure 3). 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 after setting.

Table 1 Start session for different channels (VLC cache has been set to only one second).

Chan.	Bitrate (kbps)	Start-up Delay (sec)	(*)Interv. (sec)	Higher bitrate	(**) 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

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 cannot 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. 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).

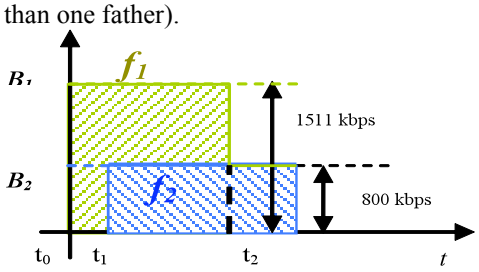

Figure 4. Traffic surplus at start-up.

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. Other client'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 helps, but it represents just a contribution.

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. Such solution does meet the user requirement and may still exploit the virtue of P2P systems. However the selection of one fixed quality can be restrictive in conditions with varying bandwidth availability (frequent event in 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 is wasteful of bandwidth. SVC coding, instead, can provide different qualities from only one stream, and 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. 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 [2]. Although SVC loses a bit in compression compared to a simulcast like approach, the latter does not fully exploits the virtue of P2P systems.

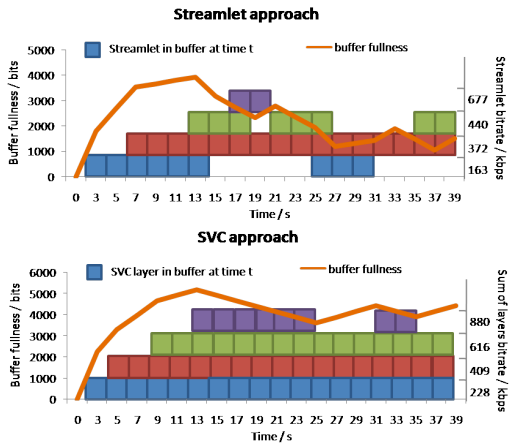

Figure 5. Switching from one quality to another: Streamlet (top) vs SVC (bottom).

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).

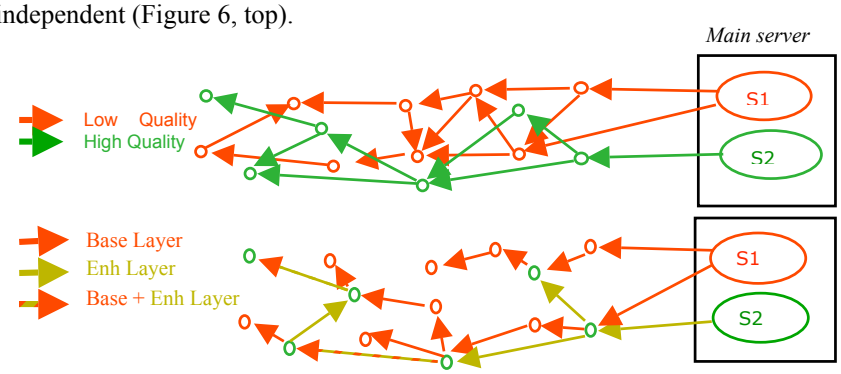

Figure 6. P2P overlay networks: independent video encoding (top) vs. SVC (bottom).

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.

A different scenario and advantages manifest themselves when the platform streams content is encoded with Multiple Description Code (MDC). The improvement, in order of relevance, is represented by resilience to missed chunks and flexibility in the description assignment. In the case of MDC based on spatial redundancy, for instance, having multiple descriptions means distributing the information of spatially closed pixels to sub-streams that will be routed through different paths and shared in different sub-overlays.

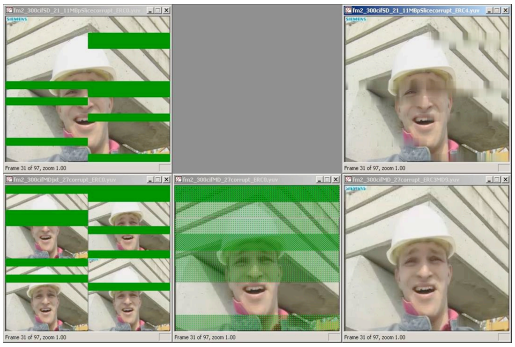

Figure 7. Loss resilience comparison between classical approach (top) and MDC (bottom).

When the client misses one description's chunk the effect at the end-user side is a picture with a few missing dots distributed over a wide and redundant spatial area. These black pixels then are covered with the information coming from the other descriptions: this is possible because the sub-stream descriptions can be decoded independently one from the other.

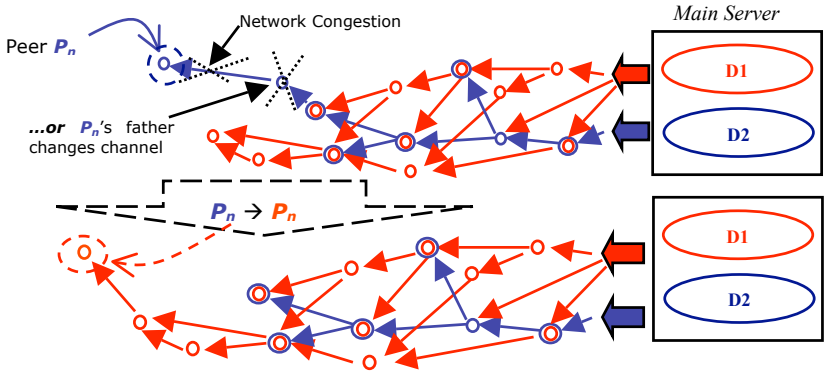

Figure 8. Flexibility of MDC in P2P streaming.

The stream loses in compression efficiency because descriptions are redundant of spatial information. On the other hand the resilience to loss outperforms the classical approach with a unique description: Figure 7 shows the effects and differences in error recovery from the end-user point of view. MDC offers also an interesting flexibility at network layer just in P2P scenarios. In Figure 8 we show an environment with a couple of description D1 and D2 and a critical (and typical) event where network flexibility is an important feature to pursue. The peer P_n, which receives the description D2, is not anymore supplied by his father at the edge of the overlay. The reason can be his father's zapping through channels or network congestion affecting the unique download link. Looking at the whole topology we observe that P_n may easily recover another description from the sub-overlay sharing D2. Since MDC is not based on hierarchical layers, the peer can change sub-overlay but still perceiving the same quality of the stream. Therefore, the peer P_n changes description's (D2 to D1) request and migrates to another sub-overlay, but still keeping the same quality of the original stream.

4. Conclusion

The aim of this work is to understand the state of the art of P2P streaming design and particularly to show the importance of the user needs in heterogeneous environments.

Our case study points out the importance of the 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. We also have described aspects of cross-layer investigation offered by using layered video coding (i.e., SVC and MDC) techniques in p2p scenarios. The optimal design incorporates a flexible implementation able to adapt to constraints efficiently and dynamically. The user point of view 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.

References

[1] www.zattoo.com

[2] T.-C. Lee, P.-C. Liu, W.-L. Shyu1, C.-Y. Wu1, *Live Video Streaming Using P2P and SVC*, IFIP/IEEE - MMNS 2008,), Samos Island, Greece, Sep 2008.

[3] A. Sentinelli, G. Marfia, S. Tewari, M. Gerla, L. Kleinrock, *Will IPTV Ride the P2P Stream?* in IEEE Communication Magazine, vol. 45, no. 6, Jun 2007.

[4] Y. H. Chu, S. G. Rao, H. Zhang, *A Case for End System Multicast*, in Proc. of ACM SIGMETRICS 2000, Santa Clara, CA, USA, Jun 2000.

[5] S. Banerjee, B. Bhattacharjee, C. Kommareddy, S*calable Application Layer Multicast*" in Proc. of SIGCOMM 2002, Pittsburgh, PA, USA, Aug 2002.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, *Splitstream: High-bandwidth Multicast in Cooperative Environments,* SOSP 2003), Bolton Landing, NY, USA, Oct 2003

[7] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, *A Measurement Study of a Large-scale P2P IPTV System*, IEEE Transactions on Multimedia, vol. 9, no. 8, Dec 2007

[8] X. Zhang, J. Liu, B. Li, T. Yum, *Coolstreaming/DONet: A Data-driven Overlay Network for Efficient Live Media Streaming*, in Proc. of the 24th IEEE INFOCOM, Miami, FL, USA, Mar 2005.

[9] X. Liao, H. Jin, Y. Liu, L. Ni, D. Deng, *Anysee: Peer-to-Peer Live Streaming*, in Proc. of the 25th IEEE INFOCOM, Barcelona, Spain, Apr 2006.

[10] V. Padmanabhan, H. J. Wang, P. A. Chou, *Supporting Heterogeneity & Congestion Control in P2P Multicast Streaming*, IPTPS 2004, San Diego, CA, USA, Feb 2004.
[11] V. Venkataraman, K. Yoshida, P. Francis, *Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicas*t, ICNP 2006, Santa Barbara, CA, USA, Nov 2006.