Assessing the Quality of Experience of SopCast

Yue Lu*, Benny Fallica and Fernando A. Kuipers

Network Architectures and Services (NAS) Group,
Faculty of Electrical Engineering,
Mathematics and Computer Science (EEMCS),
Delft University of Technology,
P.O. Box 5031, 2600 GA Delft, The Netherlands
E-mail: Y.Lu@tudelft.nl E-mail: B.Fallica@gmail.com
E-mail: F.A.Kuipers@tudelft.nl
*Corresponding author

Robert E. Kooij

TNO Information and Communication Technology,
Delft, The Netherlands
E-mail: Robert.Kooij@tno.nl

Piet Van Mieghem

Network Architectures and Services (NAS) Group,
Faculty of Electrical Engineering,
Mathematics and Computer Science (EEMCS),
Delft University of Technology,
P.O. Box 5031, 2600 GA Delft, The Netherlands
E-mail: P.F.A.VanMieghem@tudelft.nl

Abstract: Recently, there has been a growing interest in academic and commercial environments for live streaming using P2P technology. A number of new P2P digital Television (P2PTV) applications have emerged. Such P2PTV applications are developed with proprietary technologies. Their traffic characteristics and the Quality of Experience (QoE) provided by them are not well known. Therefore, investigating their mechanisms, analysing their performance, and measuring their quality are important objectives for researchers, developers and end users. In this paper, we present results from a measurement study of a BitTorrent-like P2PTV application called SopCast, using both objective and subjective measurement technologies. The results obtained in our study reveal the characteristics and important design issues of SopCast, as well as the QoE that the end users perceive.

Keywords: P2PTV; P2P digital Television; SopCast; QoE; quality of experience; measurement; PlanetLab.

Reference to this paper should be made as follows: Lu, Y., Fallica, B., Kuipers, F.A., Kooij, R.E. and Van Mieghem, P. (2009) 'Assessing the Quality of Experience of SopCast', *Int. J. Internet Protocol Technology*, Vol. 4, No. 1, pp.11–23.

Biographical notes: Yue Lu is a PhD student in the NAS group of Delft University of Technology. She obtained her BSc Degree at Huazhong University of Science and Technology (China) in 2004. She completed her MSc Degree from Delft University of Technology in April, 2006. She started her PhD research in September of 2006. Since 2004, she has been researching P2P systems and real-time internet services.

Benny Fallica has currently joined the Nokia Siemens Networks (NSN) R&D Center in Budapest, Hungary. He obtained his BSc Degree in Computer Engineering at the University of Palermo (Italy) in April, 2005. He received his MSc Degree in Computer Engineering at Delft University of Technology in December, 2007.

Fernando A. Kuipers received his MSc Degree in Electrical Engineering at the Delft University of Technology in June, 2000. He obtained his PhD Degree (cum laude) in 2004 at the same faculty. During his PhD he was a member of the DIOC (interdisciplinary research center) on the Design and Management of Infrastructures. Currently he is an Assistant Professor in the NAS group at Delft University of Technology. Since September 2006, he is also staff member of the research school for Transport, Infrastructures and Logistics (TRAIL). His work mainly focuses on algorithms for networks in general (but the internet in particular).

Robert E. Kooij received his MSc Degree in Mathematics at Delft University of Technology in 1988. He obtained his PhD Degree (cum laude) in 1993 at the same faculty. His thesis dealt with qualitative theory of non-linear Dynamical Systems. Since 1997 he is employed at TNO ICT (the former KPN Research), the largest ICT Research Institute in the Netherlands. His main research interest is Quality of Service for IP networks. Since November 2005, he is a part-time Associate Professor at the Delft University of Technology in the Network Architectures and Services section. There he works on the robustness of complex networks.

Piet Van Mieghem is Professor at the Delft University of Technology and Chairman of the Section Network Architectures and Services (NAS). His main research interests lie in new internet-like architectures for future, broadband and QoS-aware networks and in the modelling and performance analysis of network behaviour and complex infrastructures. He received a Master's and PhD in Electrical Engineering from the K.U. Leuven (Belgium) in 1987 and 1991, respectively. Before joining Delft, he worked at the Interuniversity Micro Electronic Center (IMEC) from 1987–1991. During 1993–1998, he was a member of the Alcatel Corporate Research Center in Antwerp. He was a visiting scientist at MIT (1992–1993) and, in 2005, a Visiting Professor at UCLA. Currently, he serves on the editorial board of the *IEEE/ACM Transactions on Networking*.

1 Introduction

The success of Peer-to-Peer (P2P) (BitTorrent, 2001–2008) file-sharing is undisputed. Their idea of exchanging fragments has also been applied to streaming applications over a P2P network. In recent years, many such P2P video streaming applications, e.g., CoolStreaming (2005–2008), (PPLive, 2004–2006), Tribler (Pouwelse et al., 2006) and (SopCast, 2007), have appeared and are receiving much attention. Measurements on these systems show that more than 100,000 concurrent users viewing a single channel is not uncommon. In this paper, we will investigate a P2PTV system called SopCast (Fallica et al., 2008). In order to understand the mechanisms of this BitTorrent-based P2PTV system and its performance, we will investigate by means of measurements the functionalities and the traffic characteristics of SopCast and the Quality of Experience (QoE) perceived by its end users. QoE can be measured through objective and subjective measurements. Measuring quality of user experience is important for both users and developers.

The rest of this paper is organised as follows: In Section 2 related work is discussed. In Section 3 we investigate the basic mechanisms of SopCast via conducted lab experiments. Section 4 describes measurements on a much larger network (Planetlab, 2007), in order to assess performance characteristics for end users, such as the upload and download rate and the stream quality they experience. Besides the objective measurements in Sections 3 and 4, subjective measurements are also provided in Section 5. We conclude in Section 6.

2 Related work

Hei et al. (2007) have measured PPLive via passive packet sniffing. Their measurement study focused on three important aspects of PPLive: streaming performance, workload characteristics, and overlay properties. They presented detailed session statistics, such as session duration, packet size and the correlation between them, and traffic breakdown among sessions. Start-up times and video buffer dimensions were also presented. Other work on PPLive, like Vu et al. (2007) and Wang et al. (2008), studied specific aspects of this P2P streaming system. The node degrees of popular vs. unpopular channels and the stability of nodes were investigated.

Zhang et al. (2005) and Li et al. (2008) focused on Coolstreaming, and Wu and Zhao (2008) considers UUSee.

Ali et al. (2006) evaluated the performance of both PPLive and SopCast. They collected packet traces of the systems under different conditions and analysed the data on a single host joining a system and then tuning into a channel, and collected packet traces for these cases.

Silverston and Fourmaux (2007) analysed the different traffic patterns and underlying mechanisms of several P2PTV applications. The results of this study were based on a single measurements day where two soccer games were scheduled.

Most of the above mentioned previous work was executed from a single point of observation, or from few nodes within direct access and lacks an automatic mechanism for conducting measurements. Also, the research was mainly aimed at investigating the user behaviour, without much analysis on the traffic characteristics and various mechanisms of P2PTV. Moreover, the final perception of the end user, i.e., the QoE, is not taken into account. In our opinion, it is important to investigate the QoE for P2PTV systems, since P2PTV technology can be considered a promising candidate for content distribution companies to deploy flexible and interactive TV. In this paper we perform such a study, through objective and subjective measurements, for the P2PTV application SopCast.

3 Lab experiments

In this section, we are going to investigate the basic mechanisms of SopCast by means of lab experiments.

3.1 SopCast

SopCast is a free BitTorrent-like P2PTV application, born as a student project at Fundan University in China. The bit rates of TV programmes on SopCast typically range from 250 Kb/s to 400 Kb/s with a few channels as high as 800 Kb/s. The channels can be encoded in Windows Media Video (WMV), Video file for Realplayer (RMVB), Real Media (RM), Advanced Streaming Format (ASF), and MPEG Audio Stream Layer III (MP3).

The SopCast Client has multiple choices of TV channels, each of which forms its own overlay. Each channel streams either live audio-video feeds, or loop-displayed movies according to a preset schedule. The viewer tunes into a channel of his choice and SopCast starts its own operations to retrieve the stream. After some seconds a player pops up and the stream can be seen. SopCast also allows a user to broadcast his own channel.

3.2 Measurements infrastructure

Figure 1 presents our local P2P measurements infrastructure. It is composed of standard personal computers participating in a small network. Six nodes are running the SopCast Client and the seventh one, as a SopCast broadcaster, is broadcasting a TV channel.

Traffic collection and decoding is done with Wireshark (Orebaugh et al., 2006). The nodes run Windows XP. Each node is equipped with an Intel Pentium 2.4 GHz processor, 512 MB RAM and a 10/100 FastEthernet network interface. The network interfaces are connected to a 100 Mbit switch, which is further connected through a router to the internet.

3.3 Results

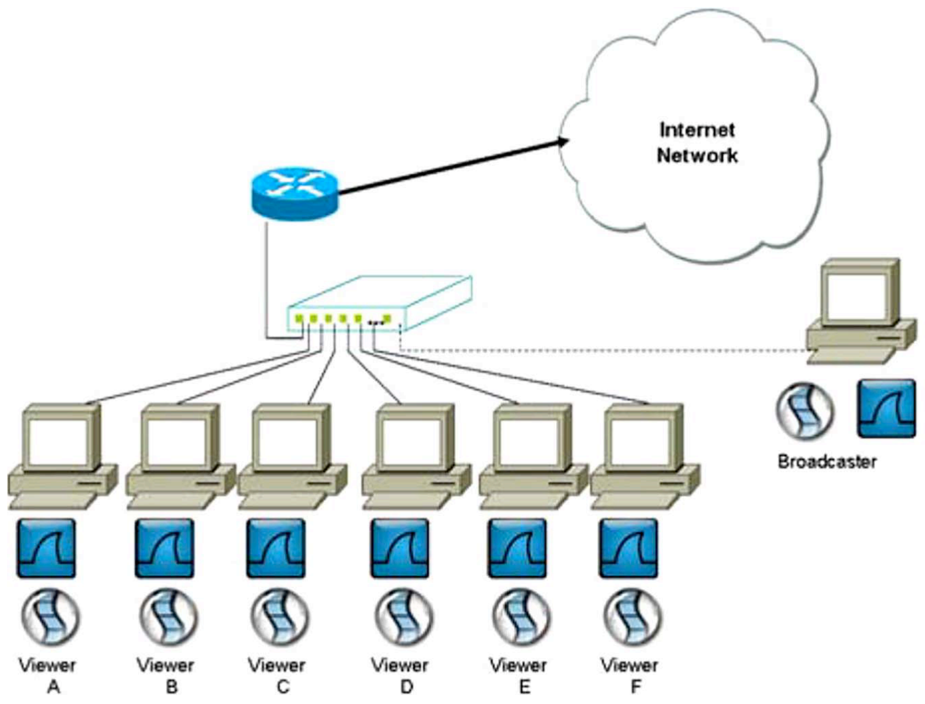
We present some observations based on our lab experiments.

3.3.1 Transport protocol

The reports of Wireshark revealed that SopCast relies on UDP traffic. We have observed two peaks in the packet size distribution: one falls in the region below 100 bytes and another one at 1320 data bytes. The small packets with less than 100 bytes are considered non-video packets, which are used for application-layer acknowledgements of data packets delivered, requests for video chunks, or initial connection establishment, and so on. We will further explain these small packets in Section 3.3.3. The bigger packets, with size approximately equal to the Maximum Transmission Unit (MTU) for IP packets over ethernet networks, are the video packets.

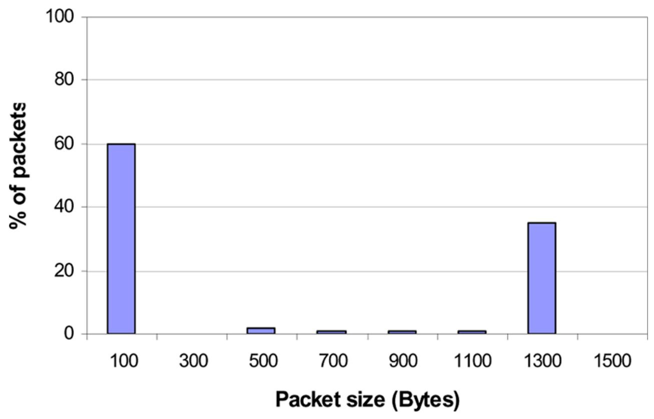
We also observed that SopCast faces a high overhead, about 60% of non-video packets vs. almost 40% of actual video data packets (see Figure 2). This was expected since the protocol works on top of UDP, which does not guarantee reliability in the way that TCP does. For time-sensitive applications, UDP is a reasonable choice, because dropped packets are considered no worse than delayed packets. However a minimum control on the status of the chunks must be kept. Since the chunks arrive out of order, a scheme is needed to keep track of the video chunks that need to be reassembled in order and buffered, and in case a chunk is missing, to retrieve it.

Figure 1 Local measurements infrastructure (see online version for colours)



Nevertheless, various small packets are exchanged among peers to keep the peer list up-to-date, to test the status of peers (e.g., there is enough bandwidth available) or to distribute the chunk availability information and the keep-alive messages. This explains the overhead in this kind of mesh-based P2PTV system.

Figure 2 SopCast packet size distribution (see online version for colours)



3.3.2 Peer exchange and architecture

When SopCast first starts, it requires some time to search for peers and subsequently it tries to download data from the active peers. We recorded two types of start-up delay: the delay from when one channel is selected until the streaming player pops up, and the delay from when the player pops up until the playback actually starts. The player pop-up delay is in general 20–30 s. This is the time for SopCast to retrieve the peer list and the first video packets. The player buffering delay is around 10–15 s, which can vary from player to player and is not related to SopCast. Therefore, the time that passes for a user to enjoy the live streaming ranges between 30 s and 45 s.

Examining the traffic generated by each node we found that the first task of each viewer node is sending out a query message to the SopCast channel server to obtain an updated channel list. This server has been identified, with an IP locator, to be located in China. After a peer selects one TV channel to watch, it sends out multiple query messages to some root servers (trackers) to retrieve an online peer list for this TV channel.

After contacting the tracker, the nodes form a randomly connected mesh that is used to deliver the content among individual peers. The content of a TV channel is divided into video chunks, each with equal size. A video chunk is delivered from a parent to a child peer. Except for the source, each peer in the overlay has multiple parents and multiple children. The delivery is performed with pull requesting by child peers, meaning that the chunks that a node has are notified periodically to the neighbours. Then each node explicitly requests the segments of interest from its neighbours according to their notification.

3.3.3 Traffic pattern

We have captured the traffic at peers and analysed how two peers communicate with each other and set up the video transmission session.

Figure 3 displays this process. First, Peer *A* requests to establish a link connection with Peer *B* (using a

Figure 3 Flow graph between two peers (see online version for colours)

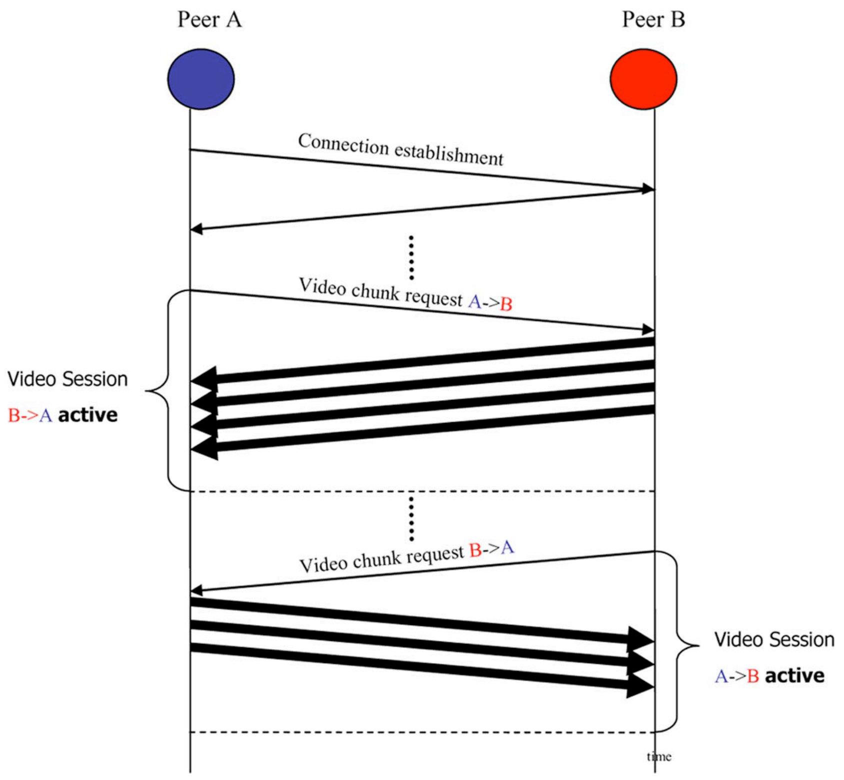
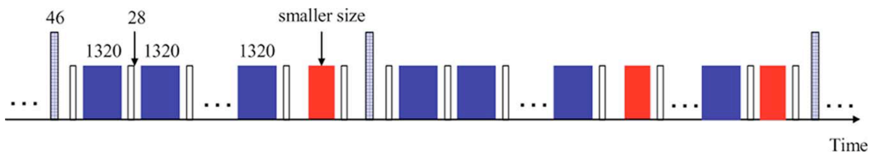


Figure 4 Traffic pattern during a video session between two peers (see online version for colours)



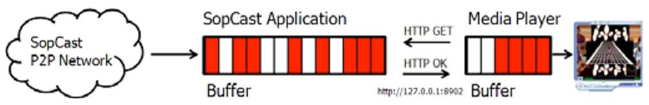
non-video packet with 52 bytes of data). After receiving the acknowledgement from Peer B, Peer A requests video chunks (using a non-video packet with 46 bytes of data) from Peer B, based on the chunk availability information. Afterwards, Peer B transmits a sequence of video packets to Peer A (shown as bold arrows in Figure 3). Similarly, Peer A can also upload the packets requested by Peer B.

In the trace, we noticed that the non-video packet with 46 data bytes are transmitted periodically. Within the transmission of two consecutive 46-byte packets, a sequence of video packets with 1320 data bytes are sent to and acknowledged (using a non-video packet with 28 data bytes) by another peer. After the 1320-byte packets sequence, there is a smaller-sized video packet (with 377, 497, 617, 1081 or 1201 data bytes) following at the end for making up a rounding size of one or multiple video chunks (we observed that a video chunk size is equal to 10 Kbytes). The SopCast traffic pattern during a video session between any two peers is shown in Figure 4.

3.3.4 Buffering techniques

Received chunks are stored in the SopCast buffer. The buffer is responsible for downloading video chunks from the network and streaming the downloaded video to a local media player. The streaming process in SopCast traverses two buffers: the SopCast buffer and the media player buffer, as shown in Figure 5.

Figure 5 The SopCast buffer (see online version for colours)



When the streaming file length in the SopCast buffer exceeds a predefined threshold, SopCast launches a media player, which downloads video content from the local Web server listening on port 8902. Most media players have built-in video buffering mechanisms. After the buffer of the media player fills up to the required level, the actual video playback starts.

The experiments presented in this section were carried out in order to understand the basic mechanisms of SopCast. In the next section we extend our measurement scenario to a global one, to learn more about the QoE of SopCast in a larger network.

4 PlanetLab experiments

In this section we present the results obtained via the PlanetLab network, using 70 PlanetLab nodes.

4.1 Measurement set-up

We have used a standard personal computer located in our campus network, as the Source Provider (SP) of a TV channel content. With the SP, we registered and broadcasted a dedicated TV channel to the SopCast network. In this channel, a video with 480×384 resolution and at 45 KB/s is continuously broadcast in a loop.

On the other hand, we have used scripts not only to remotely control 70 PlanetLab nodes (as our peers) to view the TV channel we released, but also to monitor the QoE at them.

Thus, our experiment resembles a streaming system, as shown in Figure 6.

Each of the 70 PlanetLab nodes under consideration runs the following software:

- SopCast in its Linux version, with command line control
- Tcpdump
- Perl Scripts.

Passive monitoring by its nature is limited to information acquired from the communications that are visible to the monitoring stations. By accessing all of our PlanetLab nodes, we attempt to capture data that is as complete as possible and use it for our characterisations.

We make use of traced files of this SopCast network captured during ten months (May 2007–November 2007; August 2008–October 2008). In particular, we collected the traffic logs for several one-hour intervals from the 70 peers under investigation.

4.2 Upload and download rate

Comparing the video data upload and download rates (the rate here is the average value over one hour trace), we noticed that only few nodes have higher upload rate compared to their download rate. In Figure 7 the four nodes that have higher upload than download rates have been identified as 'supernodes'.

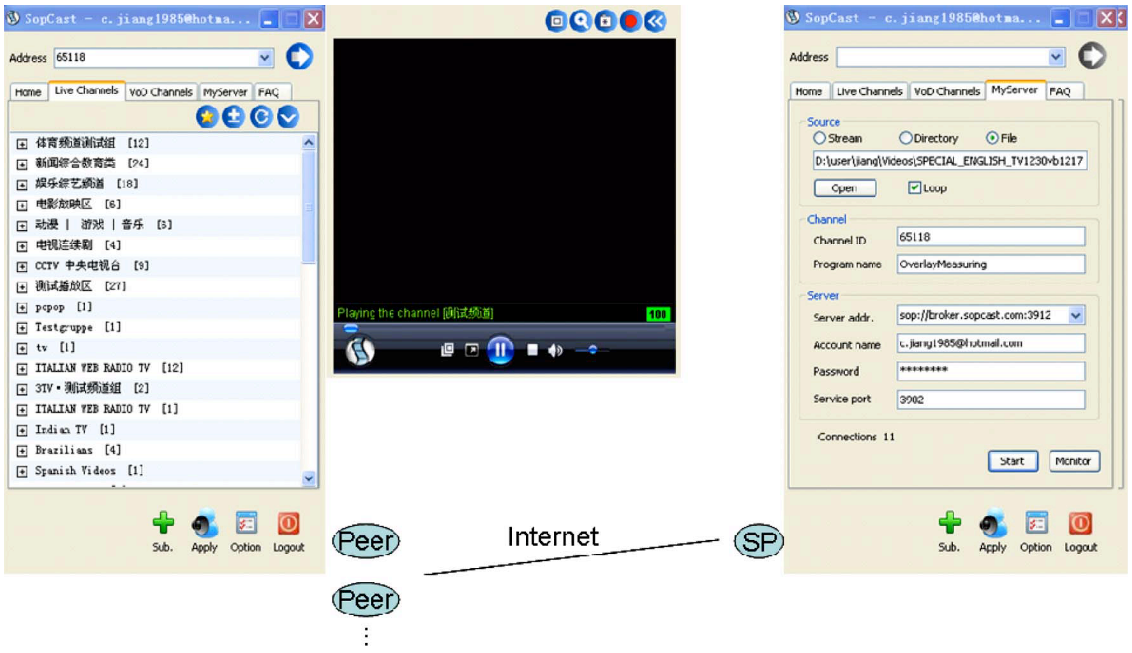
The average download rate at each node is almost the same. This suggests that the download rate at a peer seems to be confined by SopCast.

4.3 Parent's upload rate to one child

We define a parent as a peer that is uploading video packets and define one child as a receiver of video packets.

The best choice for a peer is to download from the parent who has enough 'parent upload rate' per peer

Figure 6 The SopCast player at the peer side (left); The window of the SP interface (right) (see online version for colours)



Address 65118

体育频道测试组 [12]
新闻综合教育类 [24]
娱乐综艺频道 [18]
电影放映区 [6]
动漫 | 游戏 | 音乐 [3]
电视连续剧 [4]
CCTV 中央电视台 [9]
测试播放区 [27]
pcpop [1]
Testgruppe [1]
tv [1]
ITALIAN WEB RADIO TV [12]
3TV • 测试频道组 [2]
ITALIAN WEB RADIO TV [1]
Indian TV [1]
Brazilians [4]
Spanish Videos [1]

Playing the channel [测试频道] 100

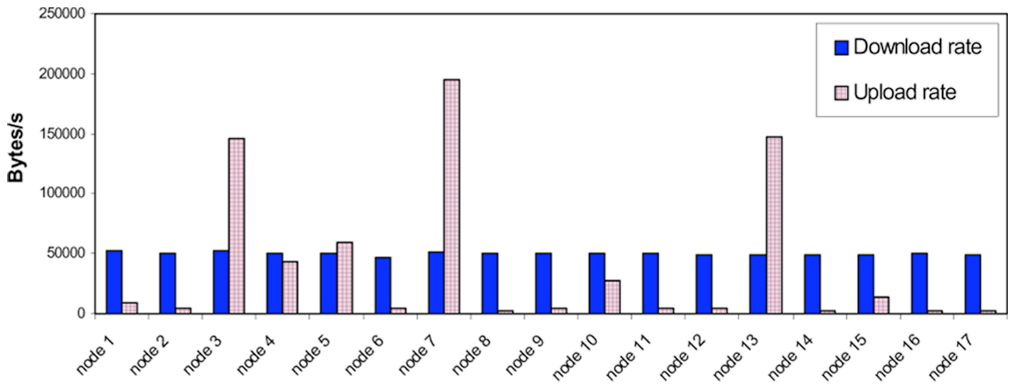
Source
Stream Directory [✓ File]
D:\user\jiang\Videos\SPECIAL_ENGLISH_TV1230vb1217
Loop ✓

Channel
Channel ID 65118
Program name OverlayMeasuring

Server
Server addr. sop://broker.sopcast.com:3912
Account name c.jiang1985@hotmail.com
Password ********
Service port 3902

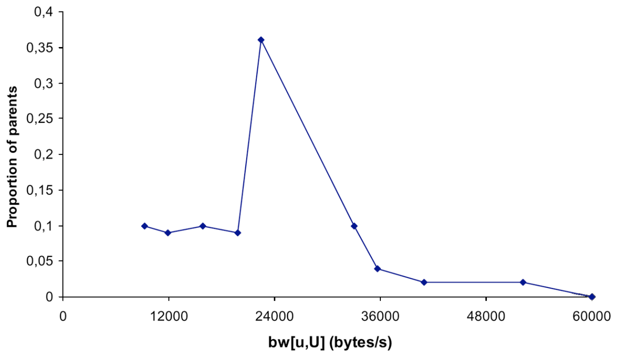
Connections 11

Figure 7 Upload and download rates of the peers at 17 of the 70 PlanetLab nodes (see online version for colours)



(the rate is the average value over one hour trace). However, from Figure 8 it can be seen that the majority of the parents keeps the same amount of upload rate per peer, which is approximately 24 KB/s. This behaviour does not change with the addition of more peers.

Figure 8 Parent's upload rate per peer when the network size is 70. u represents a parent and U represents a child of the parent (see online version for colours)



Based on the results of Figures 7 and 8, we can imagine that a parent with larger upload rate probably has more children than a parent with smaller upload rate.

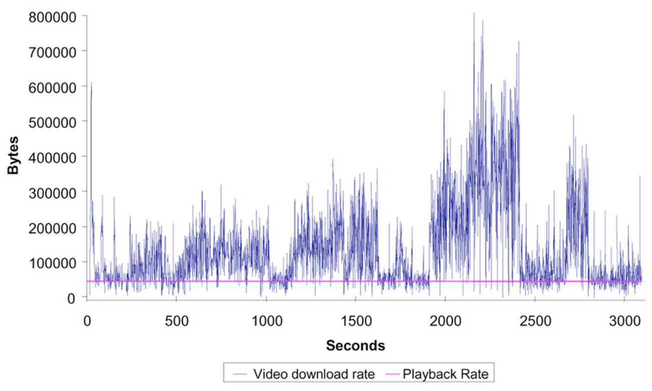
4.4 *Blocking*

In Figure 9, we consider the download rate without any buffering. We can notice the fluctuations in the download rate and we compare them with the steady playback rate of the video (45 KB/s). If the download rate is smaller than the playback rate and no data has been buffered, the end user is facing blocking or freezing of the video. In practice buffering is used, so this can be considered as a worst-case study (Lu et al., 2008).

The worst-case blocking probability is calculated counting the time t_{block}, where the download rate is smaller than the playback rate, divided by the total amount t of the observed time.

$$\Pr[\text{block}] = \frac{t_{\text{block}}}{t}.$$

Figure 9 Download rate of the video packets (see online version for colours)



From this assumption we calculated that blocking without buffering happens during 22% of the time. Such a value is too high for a smooth playback, which clearly illustrates the need for a buffer.

Figure 9 proposed a scenario without a buffer, meaning that the data will have been processed as soon as it arrives at the destination. Of course such assumption is impractical since the video packets are not arriving in order. And for the pull-architecture of SopCast a buffer is needed to map video chunks available and from that requesting the ones that are missing.

Sentinelli et al. (2007) observed that the SopCast buffer contains one minute of video. We made the assumption that the media player uses a buffer of m seconds, where m is usually smaller than 10. When an end user starts up a SopCast TV channel, basically once the SopCast buffer is full, it injects m seconds of streaming content into the media player buffer. By the time the media player consumes those m seconds of video SopCast is downloading new video packets to refill the buffer.

If the SopCast buffer fails to collect enough data to feed the media player buffer, blocking occurs.

In Figure 10 the buffer behaviour of one peer is depicted. We consider the SopCast buffer size as the streaming rate of the video (45 KB/s) times one minute (Sentinelli et al., 2007), equal to about 2700 Kbytes, which can be seen in Figure 10. We can observe that after the start-up phase, the buffer maintains stable and the playback is continuous. The average download rate for this node is with 127 KB/s far higher than the streaming rate of the video (45 KB/s). Hence, it was expected that blocking would not happen. However, due to the fluctuation of the download rate with time, the data stored in the buffer has major drops in the intervals between 1040–1150 s, 1660–1730 s, 1840–1920 s. During these drops (meaning that in these periods the data stored in the buffer is much less than the full buffer size 2700 Kbytes), end users *may* face blocking (e.g., image freezing or loss), because in the worst case, the lacked video chunks may be the ones which need to be displayed in the next m seconds.

4.5 Overall video packet loss

As mentioned before, every peer can be downloading data from other peers and at the same time be uploading data to others. For instance, we have four peers viewing our TV channel, PlanetLab nodes A, B, C and D. After analysing the trace file of node A, we know that he downloads data from nodes B, C and D during the whole trace. We can calculate how much video packets A received from B by analysing the trace file of A, as well as how much video packets B sends to A by analysing the trace file of B. Then in the video session between A and B, we can get its packet loss ratio (same for the video sessions between A and C, and between A and D). To summarise the number of packets lost in all video sessions of the receiver A, we can get the overall video packet loss ratio at node A during the whole trace. The same approach is applied to the other nodes and the distribution of the overall video packet loss ratio is plotted in Figure 11.

Figure 10 SopCast buffer content in bytes of node planetlab1.diku.dk (see online version for colours)

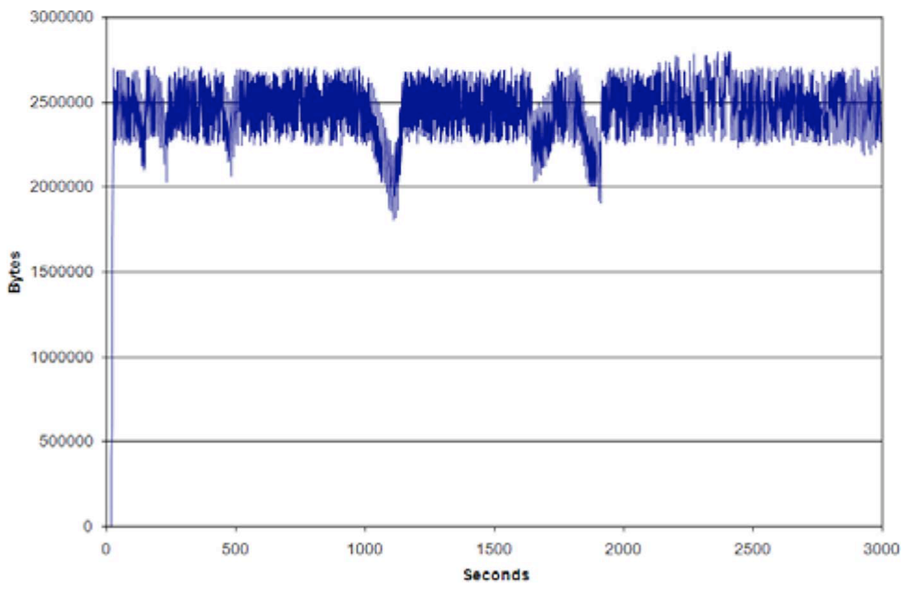
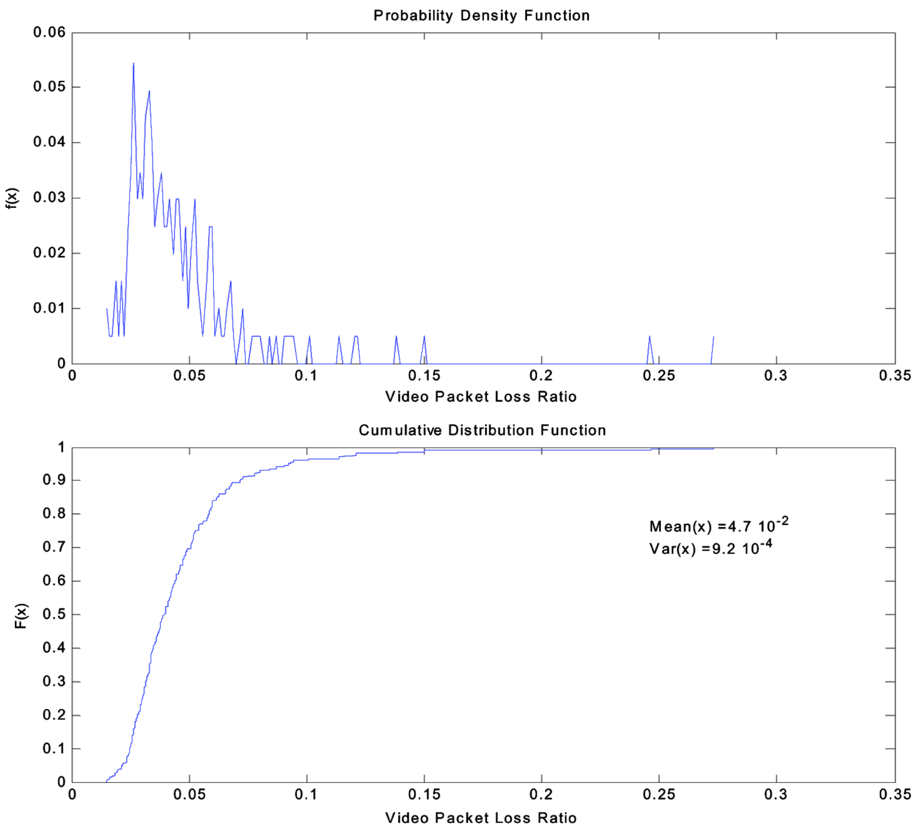


Figure 11 The overall video packet loss ratio during the whole trace (see online version for colours)



In Figure 11, the x axis represents the overall packet loss ratio during the whole trace at an end user and the y axis represents the percentage of end users in our network. The mean value of the packet loss ratio at an end user is over 4%, which is high compared to the baseline SDTV packet loss ratio requirement in IPTV of 0.4% (Agilent-Technologies, 2006). Besides, we observed that the packet loss at a peer is mainly caused in the beginning period of this peer entering this TV channel network (maybe because the connections between him and his parents are not optimised and not stable yet). However, thanks to the buffer, this high video packet loss does not have much affect on the video quality, which can be seen in Section 4.6.2.

4.6 Video quality

In this section, we assess the video quality at the end user with respect to their start-up freezing time, overall frame loss ratio, image quality and audio-video synchronisation.

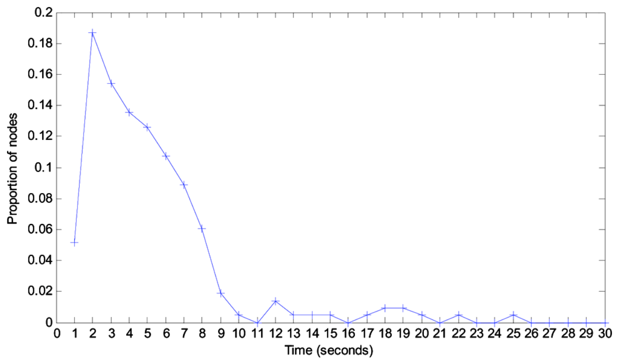
4.6.1 Start-up freezing time

Many video decoders use 'copy previous' error concealment to hide missing frames in the video stream from users. This means that in the event of not receiving a certain frame, the last correctly rendered frame is displayed on the screen, resulting in the frame freezes that are often seen in internet video playback.

Through our experiments on PlanetLab, we observed that 97.17% of nodes always first face a freezing image for a period of time at the beginning of viewing the TV channel. The reason could be that the node has just started downloading chunks from other peer nodes and the video buffer of the local SopCast webserver is created but not filled enough for the media player to access. Therefore in the beginning of viewing the TV channel, there usually exist severe frame losses with frame loss ratio approaching 100%. During this period, the media player handles the position of dropped frames by displaying the nearest good frame (the first good frame in this case) as a stagnating picture. The duration of this period (the start-up freezing time) indicates, after the user sees the first image, for how long (s)he has to wait before the video playback starts playing smoothly.

Figure 12 shows that very few end users face a start-up freezing time of more than 10 s. On average, end users see the first freezing image for 4.07 s before seeing the actual stream of the TV channel.

Figure 12 Start-up freezing time (see online version for colours)



4.6.2 Overall frame loss

The frame loss discussed here only considers lost frames, not damaged frames (frames downloaded partially with some bytes lost) after the start-up freezing phase.

In Figure 13, the x axis represents the overall frame loss ratio during the whole trace at an end user and the y axis represents the percentage of end users in our network. The mean value of the frame loss ratio is 0.82%. It means that end users could have a good video quality with low frame loss ratio after experiencing the start-up freezing time.

4.6.3 Image quality

In this section, after the start-up freezing phase of peers, we cut the received video and synchronised each frame of the captured video with the original video to get the objective Mean Opinion Score (MOS) (ITU-T, 1996), using VQM (Pinson and Wolf, 2004).

Video Quality Metric (VQM) is a software tool developed by the Institute for Telecommunication Science to objectively measure perceived video quality. It measures the perceptual effects of video impairments including blurring, jerky/unnatural motion, global noise, block distortion and colour distortion, and combines them into a single metric.

VQM takes the original video and the processed video and produces quality scores that reflect the predicted fidelity of the impaired video with reference to its undistorted counterpart. To do that, the sampled video needs to be calibrated. The calibration consists of estimating and correcting the spatial and temporal shift of the processed video sequence with respect to the original video sequence. The final score is computed using a linear combination of parameters that describe perceptual changes in video quality by comparing features extracted from the processed video with those extracted from the original video. The final score is scaled to an objective MOS value, a measure for user perceived quality, defined on a five-point scale; $5 = excellent$, $4 = good$, $3 = fair$, $2 = poor$, $1 = bad$. MOS here does not take the audio quality, zapping time, etc., into account.

We captured at selected nodes the stream retrieved from the SopCast buffer with VLC (VideoLan-Client, 2008).

We broadcasted two videos at different data rates: one at 45 KB/s (the most common data rate used in SopCast) and another one at 1 Mb/s. VQM provided the following scores per node (see Figure 14).

The minimum threshold for acceptable quality corresponds to the line $\text{MOS} = 3.5$. The average MOS scores are high for both streaming rates, only a negligible degradation has been observed. This also suggests that SopCast does not provide any kind of encoding to the broadcasted video.

4.6.4 Audio-video synchronisation

Audio-video synchronisation refers to the relative timing of sound and image portions of a television programme, or movie.

Figure 13 The overall frame loss ratio during the whole trace (see online version for colours)

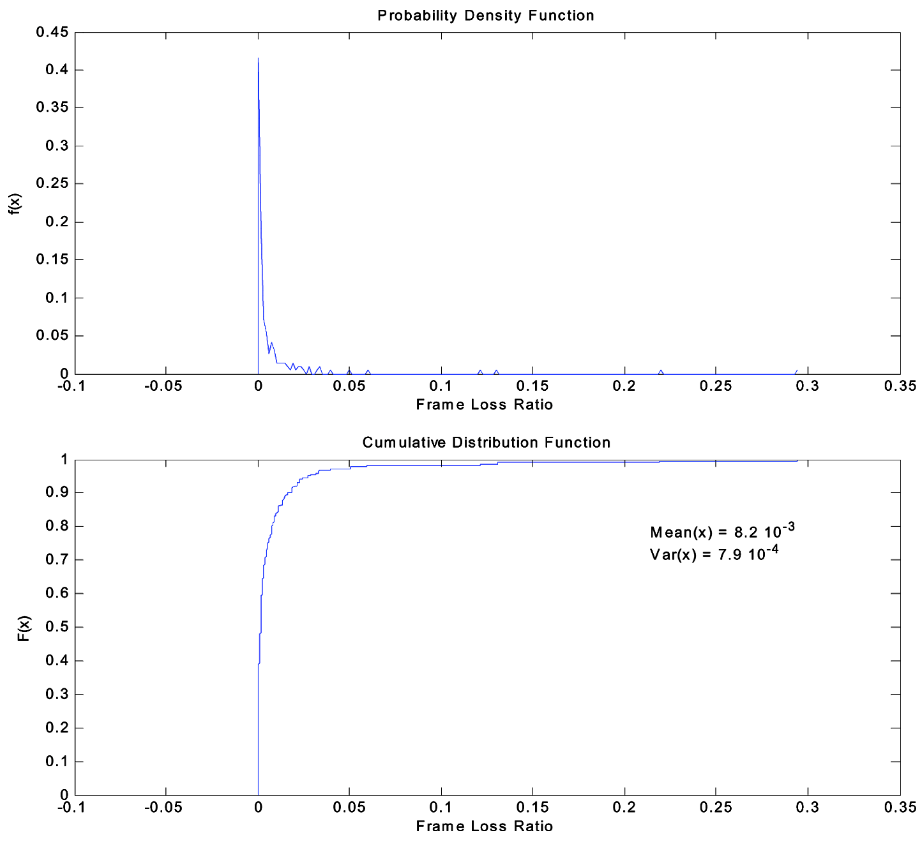
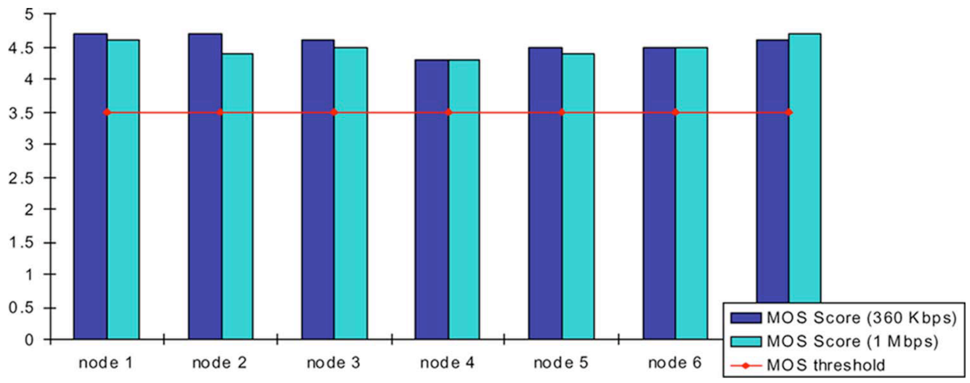


Figure 14 Objective MOS scores for the received videos (see online version for colours)



The International Telecommunications Union (ITU-R, 1998) recommendation states that the tolerance from the point of capture to the viewer/listener shall be no more than 90 ms audio leading video to 185 ms audio lagging behind video.

We decided to analyse the A/V synchronisation in SopCast with an 'artificially generated' video test sample. The test sample includes a video component and an audio component. The video component and the audio component contain a marker. The video marker displays between a first video state and a second video state, a red full screen image. Similarly, the audio waveform alternates between a first audio state and a second audio state, an audio 'beep'. The video and audio waveforms are temporally synchronised to transition from one state to another at the same time.

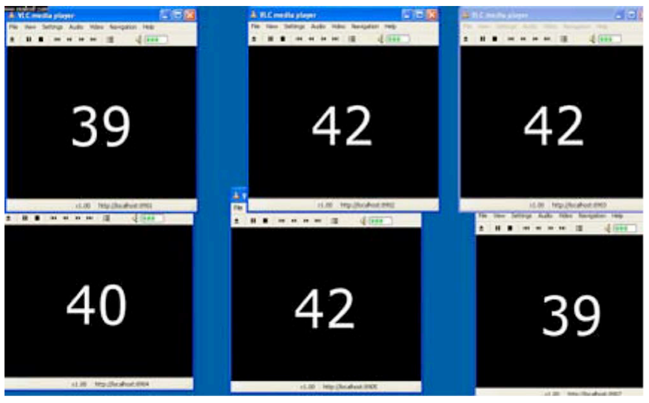
The video is broadcasted with SopCast. When the audio and video tracks were extracted and compared, it turned out that there was an average difference in time between the two tracks of about 210 ms, which exceeds the ITU recommendation. The reasons are twofold:

- We believe that the main contribution to this time shift is caused by the network. When the video is sent into the network, due to its transport protocol (UDP), some packets might get lost. Since the system is displaying in real time, a loss of a video packet can cause the decoder to adjust buffer allocations affecting the synchronisation of audio and video tracks.
- The direct digital-to-digital conversion from one (usually lossy) codec to another. We needed to convert from the original video format to the streamed one, passing through a final reconversion of the received file to extract the tracks. This (re)conversion may also have affected the synchronisation.

4.7 Peer synchronisation

While watching a football match it could be disturbing to hear the neighbours scream 'GOAL' while still watching the pre-goal action (Sentinelli et al., 2007). Such phenomena are common in P2PTV systems and are referred to as peer lags. While watching the same channel, peers' content might not be synchronised. We measured the different lag delays by injecting in the SopCast network another artificial video that mainly reproduced a timer. Each second a sequential number is shown. Since SopCast builds a webserver that feeds the player's buffer, we connected six instantiations of VLC to the webservers of the representative nodes and we gathered the visualisation on a PC, see Figure 15.

Figure 15 The video at different nodes (see online version for colours)



Clearly, some peer's content lags behind that of others. In the environment of PlanetLab, the lag went up to 3 s. In reality, the lag is expected to grow even further. Hence, we can conclude that SopCast clients are not likely to view an exactly same frame of the stream at the same time. We can say that SopCast nowadays is not yet suitable to distribute a football-like content due to the low synchronisation level among users.

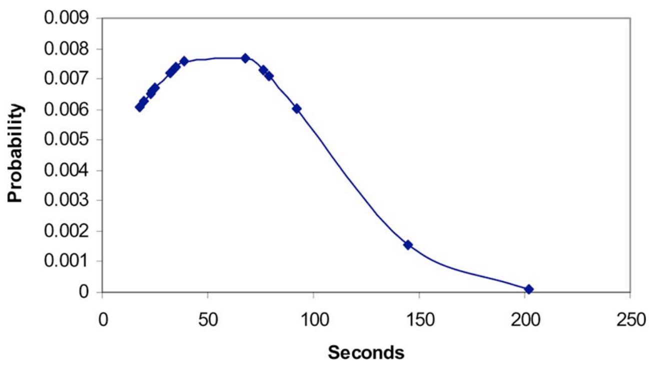
4.8 Zapping time

While watching TV a common behaviour is to change from on channel to the other, the so-called 'zapping'. If P2PTV applications want to gain popularity in the field of home entertainment it is necessary to look at the zapping performance of P2PTV applications. While for analog TV, zapping consists of scanning through different television channels or radio frequencies, in P2PTV the initial list of hosts must be retrieved, and the system tries to connect to some of the hosts to get data.

To measure the SopCast zapping time we needed to calculate the time that SopCast requires to fill its buffer and build the local web server. To do that we developed a Perl script that starts a counter when a channel is clicked and it stops when enough data to be displayed has been fetched.

We let the script run when zapping among 20 popular and less popular channels. Figure 16 shows the distribution of the zapping times. It turns out that the zapping time in SopCast is very high.

Figure 16 Distribution of zapping time (see online version for colours)



Changing channels in an analog TV network usually takes about $\frac{1}{2}$ to 1 s compared to Digital TV where zapping times of more than 2 s might be experienced. Note that according to the DSL Forum the zapping time should be limited to a maximum of 2 s (DSL-Forum, 2006). In the IPTV environment changing channels or zapping, has great importance as this is very often regarded as the most important parameter used to judge the overall quality of the network seen from the end user perspective. With an average zapping time of 50 s, SopCast (P2PTV) faces an unacceptable delay. Customers expect information being delivered to their screen as soon as possible. Hence, much improvement is needed in the start up phase of SopCast.

5 Subjective measurements

Subjective video quality is concerned with how video is perceived by a viewer and designates his or her opinion on a particular video sequence. Subjective video quality tests are quite expensive in terms of time and human resources. To evaluate the subjective video quality, a video sequence is chosen. Under typical settings of the system, the sequence is presented to the users and their opinions are collected. The opinions are scored and an average value is computed.

5.1 Approach

The following steps were used for the subjective evaluation:

- 22 persons participated in the evaluation by viewing SopCast TV channels and completing a questionnaire.
- The questionnaire contained ten questions each addressing the expected quality problems of SopCast.

The ten questions were:

- How fast was the login process?
- How long did you have to wait before seeing the stream after you started the channel?
- How long did you have to wait before seeing a stable stream?
- Was the size of the video screen satisfactory (resolution, stream bit rate)?
- During the observation period, did the video unexpectedly stop?
- Did you observe any bad frames in the video (a bad frame refers to a mosaic-like image)?
- Did you observe any freezing frames in the video (a freezing frame refers to a brief stop, say a second, in the video playback after which it resumes to a normal playback)?
- How was the voice quality (cuts, clarity, volume) of the channel?
- Were the audio and video synchronized throughout the playback time?
- Are TV channels provided by SopCast interesting and is the amount of TV channels enough?

The questionnaire used the standard MOS scale. The subjective MOS does not only consider the quality of video, but also the start-up time, the extent of the usage convenience, and the feeling about the TV channel content itself.

- Every question had a weight (the weights of the questions are also decided by end users) depending on the severity of the issue and its influence on the QoE of SopCast. Based on the weight given to each question, the overall MOS of each questionnaire was calculated as follows:

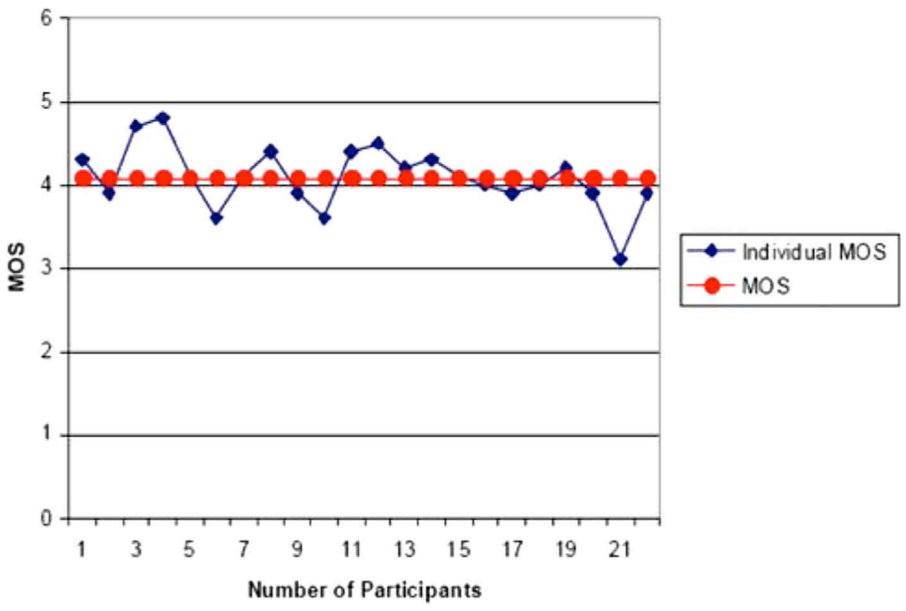
  $$\text{MOS} = \frac{\sum_{x=1}^{10} \text{Weight}_x \text{Score}_x}{\sum_{x=1}^{10} \text{Weight}_x}$$

  where Weight_x represents the weight of question x and Score_x represents the score of question x.

5.2 Result

The mean MOS over all the participants is 4.08 (see Figure 17). This means that the channel's video quality is good. The subjective MOS score is and was expected to be lower than the objective score in Section 4.6, because more measures than only video quality play a role.

Figure 17 Subjective MOS scores (see online version for colours)



6 Conclusions

The aim of this work was to understand, with a series of experiments, the behaviour of a popular P2P streaming system called SopCast. Through passive measurements, we characterised SopCast's behaviour and evaluated users' QoE.

Based on our measurement results on the traffic characteristics of SopCast, the main conclusions are:

- there is a lot of overhead in the form of non-video packets
- the average video download rate is almost the same at each peer
- peers' upload rate differs substantially, but the majority of the parents keeps the same amount of upload rate per peer
- in the worst case, a peer will face video blocking very frequently, but the situation can be much improved with the help of buffers
- overall packet loss ratio is high.

For QoE metrics, in other related works, researchers usually only look at the video quality when making claims on the QoE. However, in our work we have shown that more measures should be taken into account, such as the blocking, the audio-video synchronisation, synchronisation level among peers, the TV channel zapping time, etc. Based on our measurement results on the QoE of SopCast, the main conclusions are:

- SopCast can provide good quality video to peers: low overall frame loss ratio and high MOS scores
- audio and video for SopCast can be out-of-sync, and may even exceed the requirements from the ITU
- SopCast suffers from peer lags, i.e., peers watching the same channel might not be synchronised
- the zapping time in SopCast is extremely high.

The innovative measurement methods and scripts mentioned in our paper can also be applied to other measurement studies and for other streaming applications.

Acknowledgements

We would like to thank Siyu Tang and Javier Martin Hernandez for helping to analyse the traffic pattern of SopCast. We would also like to thank the students Vijay Sathyanarayana Rao, S.S. Gishkori, Dyonisius Dony Ariananda, Remy David, Debora Uzoamaka Ebem, Bruhtesfa E. Godana and L. Travnicek for doing the subjective measurements. This work has been supported by the Trans sector Research Academy for complex Networks and Services (TRANS) and the European Network of Excellence (NoE) CONTENT.

References

Agilent-Technologies (2006) *IPTV QoE: Understanding and Interpreting MDI Values*, White Paper, USA, p.4, URL: http://cp.literature.agilent.com/litweb/pdf/5989-5088EN.pdf

Ali, S., Mathur, A. and Zhang, H. (2006) 'Measurement of commercial peer-to-peer live video streaming', *ICST Workshop on Recent Advances in Peer-to-Peer Streaming*, Waterloo, ON, Canada.

BitTorrent (2001–2008) URL: http://www.bittorrent.com/

CoolStreaming (2005–2008) URL: http://www.coolstreaming.us/hp.php?lang=nl

DSL-Forum (2006) *Triple Play Services Quality of Experience (QoE) Requirements and Mechanisms*, Technical Report TR-126.

Fallica, B., Lu, Y., Kuipers, F.A., Kooij, R.E. and Van Mieghem, P. (2008) 'On the Quality of Experience of SopCast', *1st IEEE International Workshop on Future Multimedia Networking FMN'08*, Cardiff, Wales, UK.

Hei, X., Liang, C., Liang, J., Liu, Y. and Ross, K.W. (2007) 'A measurement study of a large-scale P2P IPTV system', *IEEE Transactions on Multimedia*, Vol. 9, No. 8, pp.1672–1687.

ITU-R (1998) 'Relative timing of sound and vision for broadcasting', *BT.1359-1*, IHS Standards Store.

ITU-T (1996) *Methods for Subjective Determination of Transmission Quality*, IHS Standards Store, pp.800–838.

Li, B., Xie, S., Qu, Y., Keung, G.Y., Lin, C., Liu, J.C. and Zhang, X.Y. (2008) 'Inside the new coolstreaming: principles, measurements and performance implications', *IEEE INFOCOM'08*, Phoenix, AZ, USA, pp.1031–1039.

Lu, Y., Kuipers, F.A., Janic, M. and Van Mieghem, P. (2008) 'E2E blocking probability of IPTV and P2PTV', *IFIP Networking*, Singapore, pp.445–456.

Orebaugh, A., Ramirez, G. and Beale, J. (2006) *Wireshark and Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security)*, Syngress Media Publishing, USA.

Pinson, M.H. and Wolf, S. (2004) 'A new standardized method for objectively measuring video quality', *IEEE Transactions on Broadcasting*, Vol. 50, No. 3, pp.312–322.

Planetlab (2007) URL: http://www.planet-lab.org/

Pouwelse, J.A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Losup, A., Epema, D.H.J., Reinders, M.J.T, Van Steen, M.R. and Sips, H.J. (2006) 'Tribler: a social based peer to peer system', *5th International Workshop on Peer-to-Peer Systems IPTPS*, Santa Barbara, CA, pp.16–21.

PPLive (2004–2006) URL: http://www.pplive.com/en/index.html

Sentinelli, A., Marfia, G., Gerla, M., Kleinrock, L. and Tewari, S. (2007) 'Will IPTV ride the peer-to-peer stream?', *Communications Magazine*, Vol. 45, No. 6, pp.86–92.

Silverston, T. and Fourmaux, O. (2007) 'Measuring P2P IPTV systems', *Network and Operating Systems Support for Digital Audio and Video NOSSDAV*, Urbana-Champaign, IL, USA.

SopCast (2007) URL: http://www.sopcast.org/

VideoLan-Client (2008) URL: http://www.videolan.org

Vu, L., Gupta, I., Liang, J. and Nahrstedt, J. (2007) 'Measurement of a large-scale overlay for multimedia streaming', *16th International Symposium on High Performance Distributed Computing*, Monterey, CA, pp.241–242.

Wang, F., Liu, J. and Xiong, Y. (2008) 'Stable peers: existence, importance, and application in peer-to-peer live video streaming', *IEEE Infocom'08*, Phoenix, AZ, USA, pp.1364–1372.

Wu, C., Li, B. and Zhao, S. (2008) 'Multi-channel live P2P streaming: refocusing on servers', *IEEE INFOCOM'08*, Phoenix, AZ, USA, pp.1355–1363.

Zhang, X., Liu, J., Li, B. and Yum, T.S.P. (2005) 'Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming', *24th IEEE INFOCOM*, Miami, FL, USA, pp.2102–2111.