On the Quality of Experience of SopCast

Benny Fallica[1], Yue Lu*[1], Fernando Kuipers[1], Rob Kooij[2], and Piet Van Mieghem[1]
[1] Delft University of Technology, b.fallica@gmail.com,
{Y.Lu, F.A.Kuipers, P.F.A.VanMieghem}@tudelft.nl,
[2] TNO Information and Communication Technology, Robert.Kooij@tno.nl
*Corresponding author

***Abstract*— Peer-to-Peer (P2P) file sharing has become immensely popular in the Internet. Recently, there has been a growing interest in academic and commercial enviornments 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 and the Quality of Experience (QoE) provided by them is not well known. Therefore, investigating their mechanisms, analyzing their performance, and measuring their quality are important for researchers, operators and end users. In this paper, we present results from a measurement study of a P2PTV application called SopCast, using both objective and subjective measurement technologies. The results obtained in our study reveal important design issues of SopCast and the QoE that the end users perceive.**

I. Introduction

The success of peer-to-peer (P2P) BitTorrent[1] file-sharing is undisputed. Their idea of exchanging fragments has also been applied to streaming applications over a peer-to-peer network. In recent years, a lot of such peer-to-peer video streaming applications, e.g. CoolStreaming [1], PPLive [2] and SopCast[2], 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. 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 characteristics of SopCast and the Quality of Experience (QoE) perceived by its end users. Measuring quality of user experience is important for both users and developers. QoE can be measured through objective and subjective measurements.

The rest of this paper is organized as follows: In Section II related work is discussed. In Section III we investigate the basic mechanisms of SopCast via conducted lab experiments. Section IV describes measurements on a much larger network, PlanetLab[3], in order to assess performance characteristics for end users, such as e.g. the upload and download rate and the stream quality they experience. Besides the objective measurements in Sections III and IV, subjective measurements are also provided in Section V. We conclude in Section VI.

[1]http://www.bittorrent.com/
[2]http://www.sopcast.org/
[3]http://www.planet-lab.org/

II. Related work

There are numerous P2PTV applications available: e.g. SopCast, CoolStreaming/DONET [1], Joost[4], PPlive [2], PPstream[5], TVAnts [3], Tribler [4], etc. However, only few measurement studies were performed on P2PTV.

Hei *et al.* [2] 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.

Ali *et al.* [5] evaluated the performance of both PPLive and SopCast. They collected packet traces of the systems under different conditions and analyzed 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 [6] analyzed the different traffic patterns and underlying mechanisms of several P2PTV applications. During the 2006 FIFA World Cup, they collected packet traces from PPLive, PPStream, SopCast and TVAnts. This work differs from [2] by the number of applications studied and the followed comparative approach. Results in this study are based on a single day where two soccer games were scheduled. The measured download traffic indicates that the applications use different mechanisms to obtain the video and in addition they maintain a different peer neighborhood.

Most of the previous work is executed from a single point of observation [6], or from few nodes within direct access [5] and lacks an automatic mechanism for conducting measurements. Also, the final perception of the end user, i.e. the Quality of Experience, is not taken into account. In our opinion, it is important to investigate the Quality of Experience 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.

III. Lab Experiments

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

[4]http://www.joost.com/
[5]http://www.ppstream.cn/ (in Chinese)

A. SopCast

SopCast is a free P2PTV application, born as a student project at Fundan University in China. The bit rates of TV programs on SopCast typically range from 250 Kbps to 400 Kbps with a few channels as high as 800 Kbps. The channels are encoded in two formats: Windows Media Video (WMV) or Real Video (RMVB).

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.

B. Measurement 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.

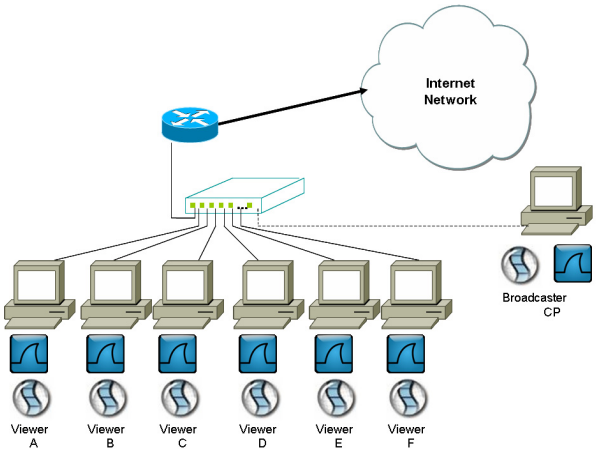

Fig. 1. Local measurements infrastructure.

Traffic collection and decoding is done with Wireshark [7]. The nodes run Windows XP. Each node is equipped with an Intel Pentium 2.4GHz processor, 512 MB RAM and a 10/100 FastEthernet network interface. The network interfaces are connected to a 100Mbit switch, which is further connected through a router to the internet.

C. Results

We present some observations based on the lab experiments.

1) Transport protocol: The reports of Wireshark revealed that SopCast relies on UDP traffic. In Figure 2 we have plotted the packet sizes in a histogram. We can observe two peaks: one falls in the region below 100 bytes and another one at 1320 bytes. The small packets are application-layer acknowledgments of data packets sent and received. The bigger packets, with size approximately equal to the Maximum Transmission Unit (MTU) for IP packets over Ethernet networks, are the video packets.

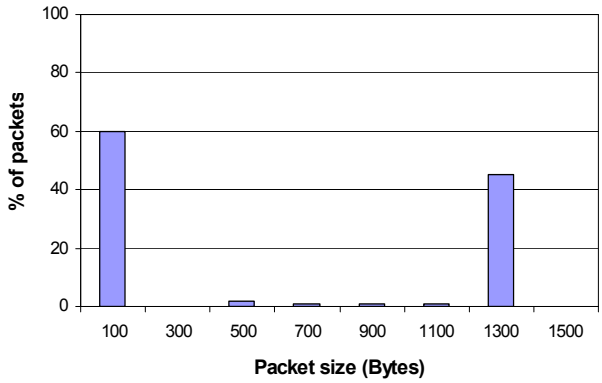

Fig. 2. Histogram of the size of the captured packets.

Figure 2 indicates that SopCast faces a high overhead, about 60% of signaling packets versus 40% of actual video data packets. This was expected since the protocol works on top of UDP, which does not guarantee reliability in the way that TCP does. The packets may arrive out of order, appear duplicated, or go missing without notice. For time-sensitive applications, UDP is the most reasonable choice, because dropped packets are preferable to 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. This explains the overhead.

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 to 30 seconds. This is the time for SopCast to retrieve the peer list and the first video packets. The player buffering delay is around 10 to 15 seconds, 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 and 45 seconds.

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. Before a peer actually starts to watch a channel, it does not exchange data with other SopCast peers. After a peer selects one channel to watch, it sends out multiple query messages to some root servers to retrieve an online peer list for this channel. Peers are identified by their IP addresses and port numbers on the list. Upon receiving a peer list, the SopCast client sends out probes to peers on the list to find active peers for the channel of interest. Some active peers may

also return their own peer lists, helping the initial peer to find more peers. Peers then share video chunks with each other.

After contacting the tracker, the nodes form a randomly connected mesh that is used to deliver the content among individual peers. Data 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 neighbors. Then each node explicitly requests the segments of interest from its neighbors according to their notification. Each peer receives content from all its parents and provides content to all of its child peers in the overlay.

3) 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 3.

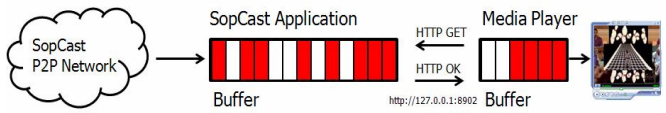

Fig. 3. The SopCast buffer

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 over the Internet.

IV. PlanetLab Experiments

In this section we present the results obtained via the PlanetLab network.

A. Measurement set-up

We used scripts not only to remotely control SopCast at 70 PlanetLab nodes, but also to analyze the QoE at them.

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

1) SopCast in its Linux version, with command line control.
2) Tcpdump.
3) 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 characterizations.

We make use of traced files of this SopCast network captured during 6 months (May 2007 – Nov. 2007). In particular, we collected the traffic logs for several one-hour intervals from the 70 peers under investigation.

B. Upload and Download rate

Comparing the upload and download rates, we noticed that only few nodes have higher upload rate compared to their download rate. In Figure 4 the four nodes that have higher upload than download rates have been identified as "supernodes".

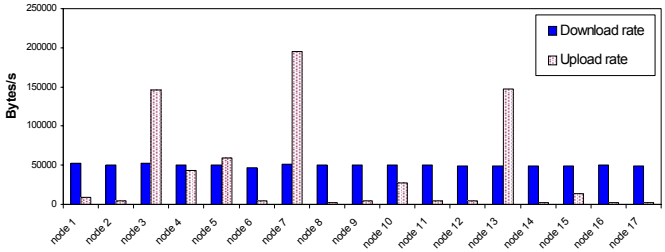

Fig. 4. Upload and download rates of the peers at 17 of the 70 PlanetLab nodes.

C. Parent's upload rate to one of his children

The best choice for a peer is to download from the parent which has enough "parent upload rate" per peer. However, from Figure 5 it can be seen that the majority of the parents keeps the same amount of upload rate per peer, which is about 24000 bytes/s. This behavior does not change with the addition of more peers.

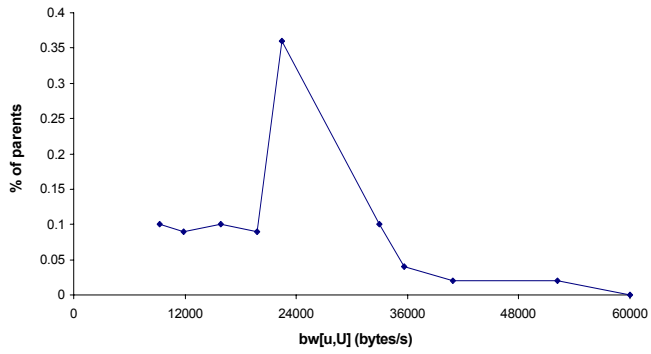

Fig. 5. Parent's upload rate per peer when the network size is 70. In the figure, u represents a parent and U represents a child of the parent.

The curve in Figure 5 slightly resembles a Gaussian distribution. Based on the results of Figures 4 and 5, we can imagine that a parent with larger upload rate probably has more children than a parent with smaller upload rate.

D. Blocking

Sentinelli *et al.* [8] observed that the SopCast buffer contains one minute of video. We made the assumption that the media player uses a buffer of m seconds, which 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. The viewer notices this blocking when the SopCast buffer drains out, meaning that no data is present for the media player to process.

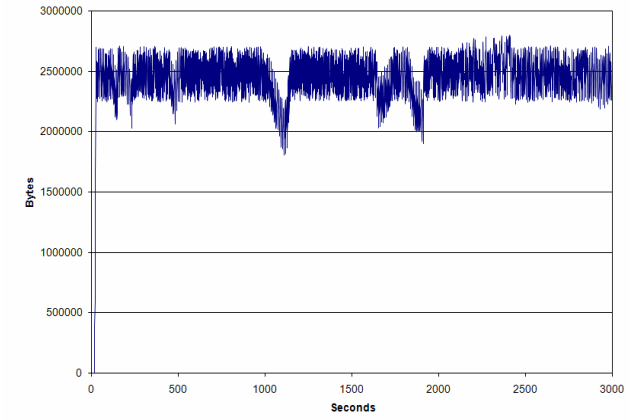

Fig. 6. Buffer content in bytes of node planetlab1.diku.dk.

In Figure 6 the buffer behavior is depicted. 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 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, end users may face blocking (e.g., image freezing or loss).

E. Video Quality

VQM (Video Quality Metric) [9] is a software tool developed by ITS (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 color 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 consist 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 Mean Opinion Score (MOS), a measure for user perceived quality, defined on a five-point scale; 5 = *excellent*, 4 = *good*, 3 = *fair*, 2 = *poor*, 1 = *bad* [10]. 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[6].

We broadcasted two videos at different data rates: one at 300 Kbps (the most common data rate used in SopCast) and another one at 1 Mbps. VQM provided the following scores per node (see Figure 7):

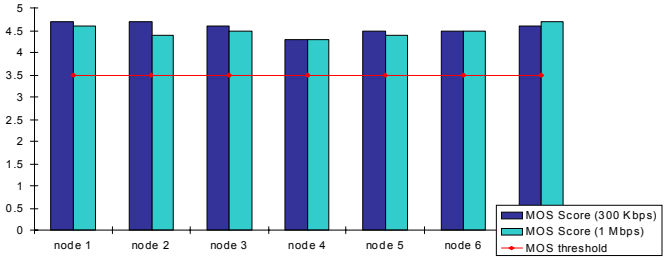

Fig. 7. VQM score for the received videos.

The minimum threshold for acceptable quality corresponds to the line MOS = 3.5. The VQM scores are high for both data rates, only a negligible degradation has been observed. This suggests that SopCast does not provide any kind of encoding to the broadcasted video. Effects of high playback rates are latency in the video and possible blocking of the image. The SopCast protocol, in order to deliver the video, prefers to face a video blocking emptying the buffer instead of degrading the image quality.

Since SopCast relies on UDP traffic, it is expected that some frames might get lost. However, SopCast did not experience a degradation in the image quality, meaning that the overall experience remained good.

F. Audio-Video Synchronization

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

The International Telecommunications Union (ITU) [11] recommendation states that the tolerance from the point of capture to the viewer/listener shall be no more than 90 msec audio leading video to 185 msec audio lagging behind video.

We decided to analyze the A/V synchronization 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

[6]VideoLan Client http://www.videolan.org

waveforms are temporally synchronized 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:

1) 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. Issues such as packet loss, data errors, and packet reordering can affect delay. Since the system is displaying in real time, a loss of a video packet can cause the decoder to adjust buffer allocations affecting the synchronization of audio and video tracks.
2) 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 might also have affected the synchronization.

G. Peer Synchronization

While watching a football match it could be disturbing to hear the neighbors scream "GOAL" while still watching the pre-goal action. Such phenomena are common in P2PTV systems and are referred to as peer lags. While watching the same channel, peers' content might not be synchronized. 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 6 instantiations of VLC to the webservers of the representative nodes and we gathered the visualization on a PC, see Figure 8.

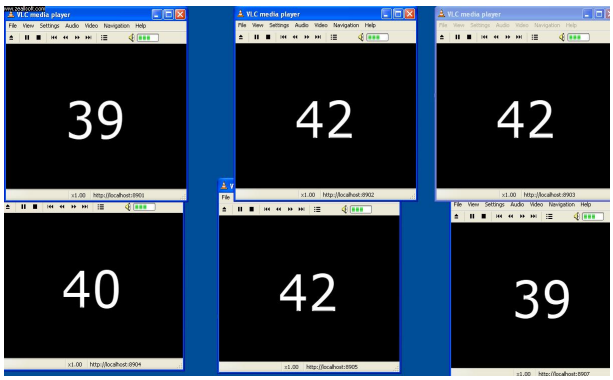

Fig. 8. Visualization of the video at different nodes.

Clearly, some peer's content lags behind that of others. In the controlled environment of PlanetLab the delay is about 3 seconds. However, in reality, with many more peers, the lag is expected to grow even further.

H. Zapping Time

While watching TV a common behavior 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 9 shows the distribution of the zapping times. It turns out that the zapping time in SopCast is very high.

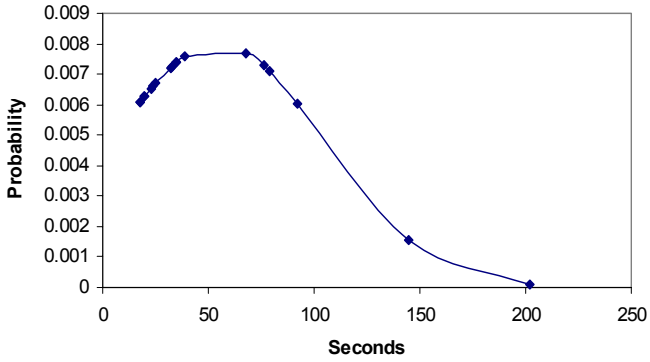

Fig. 9. Distribution of zapping time.

Changing channels in an analog TV network usually takes about $\frac{1}{2}$ to 1 second compared to IPTV where zapping times of more than 2 seconds might be experienced. Note that according to the DSL Forum the zapping time should be limited to a maximum of 2 seconds [12].

With an average zapping time of 50 seconds, SopCast faces an unacceptable delay. Hence, a huge improvement is needed with respect to the zapping performance with SopCast.

V. 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.

A. Approach

The following steps were used for the subjective evaluation.

1) A questionnaire containing 10 questions each addressing the expected quality problems of SopCast was set up. The questionnaire uses the standard (1-5) MOS scales. 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 has a weight[7] depending on the severity of the issue and its influence on the QoE of SopCast.

2) Based on the weight given to each question, the overall MOS of each questionnaire is calculated using the following formula:

$$MOS = \frac{\sum_{x=1}^{10} Weight_x Score_x}{\sum_{x=1}^{10} Weight_x}$$

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

3) 22 questionnaires were completed by 22 participants.

B. Result

The mean MOS over all the participants is 4.08, see Fig 10. This means that the channel's video quality has been good. The subjective MOS score is and was expected to be lower than the objective score in Section IV-E, because more measures than only video quality played a role.

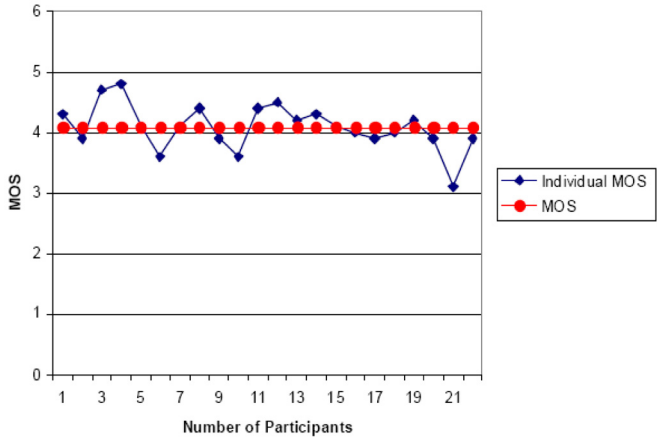

Fig. 10. Subjective MOS

VI. Conclusions

The aim of this work was to understand, with a series of experiments, the behavior of a popular P2P streaming system called SopCast. Through passive measurements, we characterized SopCast's behavior and user experience. We performed our experiments in a controlled environment and evaluated the QoE.

Based on the measurement results on PlanetLab, the main conclusions are:

- SopCast can provide good quality video to peers broadcasting from a PC.
- SopCast, despite using UDP, shows no impairment of the received video with respect to the original.
- 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 synchronized.
- The zapping time in SopCast is extremely high, on average 50 seconds.

[7]The weights of the questions are also decided by end users.

Acknowledgements

We would like to thank the students Vijay Sathyanarayana Rao, S.S. Gishkori, Dyonisius Dony Arianandа, Remy David, Debora Uzoamaka Ebem, Bruhtesfa E. Godana and L. Travnicek for doing the subjective measurements.

This work has been partially supported by European Union CONTENT NoE (FP6-IST-038423)[8]. The work of Yue Lu was supported by the Dutch Research Delta.

[8]http://www.ist-content.eu

References

[1] X. Zhang, J. Liu, B. Li, and TS. P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media streaming", Proc. of IEEE INFOCOM, Mar, 2005, vol.3, pp.2102-2111.

[2] X. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, "A Measurement Study of a large-Scale P2P IPTV System", IEEE Transactions on Multimedia, Oct. 2007.

[3] T. Silverston, O. Fourmaux, "P2P IPTV Measurement: A Case Study of TVAnts", in procedings of student workshop of Conference on Future Networking Technologies (CONEXT'06), December 2006.

[4] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker1, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M. van Steen1, H.J. Sips, "Tribler: A social based Peer to Peer system", 5th Int'l Workshop on Peer-to-Peer Systems (IPTPS).

[5] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming", In proc. of ICST Workshop on Recent Advances in Peer-to-Peer streaming, 2006.

[6] T. Silverston, O. Fourmaux, "Measuring P2P IPTV Systems", in procedings of Network & Operating Systems Support for Digital Audio & Video (NOSSDAV'07), June 2007.

[7] A. Orebaugh, G. Ramirez, J. Burke, and L. Pesce, "Wireshark & ethereal network protocol analyzer toolkit (jay beale's open source security)", Syngress Publishing, 2006.

[8] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, S. Tewari, "Will IPTV ride the peer-to-peer stream?", Communications Magazine, IEEE, Volume: 45, Issue: 6, On page(s): 86-92, June 2007.

[9] M. H. Pinson and S. Wolf, "A new standardized method for objectively measuring video quality", IEEE Transactions on broadcasting 50 (2004), no. 3, 312-322.

[10] ITU-T Rec. P.800, "Methods for Subjective Determination of Transmission Quality", 1996.

[11] The International Telecommunications Union (ITU) BT.1359-1 (11/98) Relative timing of sound and vision for broadcasting.

[12] DSL Forum, "Triple Play Services Quality of Experience (QoE) Requirements and Mechanisms", Technical Report TR-126, 13 Dec., 2006.