SKYPE RESILIENCE TO HIGH MOTION VIDEOS

GEORGE EXARCHAKOS∗†‡, LUCA DRUDA†§, VLADO MENKOVSKI∗¶, PAOLO BELLAVISTA∥§ and ANTONIO LIOTTA∗∗

∗Electrical Engineering, Eindhoven University of Technology
P.O. Box 513, Eindhoven, 5600 MB, The Netherlands
†DEIS, University of Bologna, Viale Risorgimento 2
Bologna, 40136, Italy
‡g.exarchakos@tue.nl
§luca.druda@studio.unibo.it
¶v.menkovski@tue.nl
∥paolo.bellavista@unibo.it
∗∗a.liotta@tue.nl

Received 20 August 2012
Revised 21 December 2012
Accepted 28 January 2013
Published 31 May 2013

Skype is one of the most popular video call services in the current Internet world. One of its strengths is the use of an adaptive mechanism to match the constraints of the underlying network. This work is focused on how this mechanism can maximize the video quality as perceived by the viewers using objective assessment methods. We built a testbed to stream certain video sequences through Skype between two clients over impaired communication channels. Original and recorded videos were compared to assess the achieved quality. Extensive experimentation has shown that Skype has problems when transmitting high motion videos and especially complex videos with frequent interchange between frames of low and high temporal information. The results suggest that random packet loss intensifies quality degradation for those videos more than packet loss bursts or jitter.

Keywords: Skype; video quality; network impairments; adaptive streaming.

AMS Subject Classification: 68M10, 68M12, 68M15

1. Introduction

Skype is one of the well-known applications that has guided the evolution of real-time video streaming and become one of the most used software in everyday life. As a primary video conferencing application, Skype assumes certain characteristics of the delivered video to optimize its perceived quality. However in the last few years
and with the recent release of SkypeKit, many new Skype video-enabled devices came out especially in the mobile world. This has forced a change to the traditional recording, streaming and receiving settings allowed for a wide range of network and content dynamics.

Video calls are not based on static “chatting” anymore. Mobile devices have opened new possibilities and can be used in several scenarios. For instance, lecture streaming or one-to-one mobile video conferences exhibit more dynamics as both caller and callee might be on the move. Most of these cases are different from “head&shoulder” only content. Therefore, Skype needs to optimize its video streaming engine to cover more video types.

Since Skype protocol is proprietary, most of the studies so far have tried to characterize its traffic and to reverse engineer its protocol. However, questions related to the behavior of Skype, especially on quality as perceived by users, remain unanswered. The motivation of our work is the design of a mechanism that estimates the perceived cost of network conditions on Skype video delivery. To this extend, we try to assess in an objective way the impact of network impairments on the perceived quality of a Skype video call. Traditional video streaming schemes lack the necessary flexibility and adaptivity that Skype tries to achieve at the edge of a network.

Our contribution lies on the testbed we developed and the objective video quality analysis we carried out on input videos. We streamed raw video files with Skype via an impaired channel. Transmitted videos were recorded at the receiver side and aligned with the original input samples. Finally, we carried out an analysis of the videos with objective quality of experience metrics such as SSIM. The results show the weaknesses of Skype protocol with high motion and especially with “mixed-motion” videos and fast changes to network channel conditions.

This paper is organized into six sections. After the introduction and related work, Sec. 3 describes the setup of the testbed we built to run the objective video quality measurements on Skype video calls. We carry on with the design of the actual experiments in Sec. 4. Section 5 hosts the results and a discussion on them. Overview of the conclusions and future work plans are included in Sec. 6.

2. Related Work

The first works on Skype were based on the analysis of its architecture and mainly focused on how it dealt with NAT and Firewalls. Subsequently, researchers focused their attention over the identification and classification of Skype traffic. Bonfiglio et al. in 2006 presented two classifiers to reveal Skype traffic from aggregate streams of packets\(^1\): the first approach exploited the statistical properties of message content to let patterns and structures naturally emerge while the second one used Naive Bayesian techniques to match the stochastic characteristics of voice traffic generated

Skype Resilience to High Motion Videos

by Skype. Beyond the identification of Skype calls, they also explained in details basic concepts about message formats in Skype transmission. The classification methods were further extended in Ref. 2.

In Ref. 3, authors have characterized the traffic generated by voice and video calls, by observing simulations in terms of bit rate, inter-packet gap and packet size. To this end they indicate three parameters as those determining the characteristics of generated traffic: the message framing time ($\Delta T$), the rate used by the source and the Redundancy Factor (RF) that characterizes the Forwarding Error Correction (FEC) mechanism used by Skype. Authors also showed that Skype calls always have an initial phase in which the network conditions are unknown and for this reason, an aggressive setting of the RF is made assuming the presence of a bad network. Finally they explore some characteristics of Skype users’ behavior like workload, call duration and destination, peer-life and signaling that use classifiers based on Naive Bayes algorithm and Chi-Square statistical test.

Most of the recent works are addressed more on the user experience analysis rather than studying Skype internal structure and media flows characterization. In Ref. 4, a QoE-centric voice call study of Skype FEC mechanism has been done trying to understand if there is margin to improve user satisfaction. Analyzing voice calls in both PC-to-PC and PC-to-PSTN scenarios pointed out that Skype changes the proportion of packets with redundant FEC data based on the network loss rate and it never considers the difference of each codec.

The study of FEC mechanism related to voice calls and dummy networks is of great interest in mobile networks especially after the UMTS operator has begun to offer large data rates that theoretically is capable to support VoIP traffic. In Ref. 5, authors analyzed the achieved quality of IP-based voice call using Skype both under real UMTS network and testbed environment to reproduce rate control mechanism and network variability. The measurements of the QoE are done through MOS values while QoS is measured with typical metrics like throughput, jitter and packet losses. They ended saying the capacity offered by UMTS is sufficient to make mobile VoIP calls possible, however the MOS values are worse than in a wired environment due to the jitter.

Up till now, there are few works focused on video calls analysis and most of them try to investigate the traffic behavior. De Cicco and Mascolo in Ref. 6 take a step beyond Skype Voice-over-IP traffic characterization and provide a mathematical model for the congestion control that Skype uses for voice calls.

In Ref. 7, authors have built an experimental testbed through Linux machines using iptables\(^b\) for traffic measurements and have developed a software called Skype Measurement Lab to inject a desired video from the file system into Skype. The aim of that work was to discover to what extent Skype is able to throttle its sending rate to match the available bandwidth while preserve resource for co-existing best-effort TCP traffic. The results showed once again that every time a large loss event

\(^b\)http://www.netfilter.org.
happens when the packet size doubles; meaning that Skype is applying the FEC algorithm.

Finally, Jing Zhu in 2011, published the results of an experiment on Skype video call quality. The aim of the paper was to study the stream behavior over a WiMAX network. They streamed a video through a typical WiMAX network configuration and recorded it at the receiver side. To align reference and recorded video, the author described a cross correlation method. Once videos were aligned, the quality was assessed based on effective frame rate and mean opinion score.

While many studies tried to provide experimental or formal characterization of Skype streams, only Ref. 8 moved to the objective assessment of delivered video. While Ref. 8 tries to assess the Skype video quality over WiMAX network pipes, in this work we try to understand how exactly changes on network pipes may affect the delivered quality as perceived by users. We devise a more accurate (non probabilistic) method for aligning sample and recorded videos and assess the quality based on more accurate objective metrics. The second dimension we will study is the use of Skype as a streaming tool with videos beyond the simple head&shoulder types. We try to understand to which extend Skype may employ beyond simple video calling.

3. Experimental Setup

Given our motivation of assessing the resilience of Skype as a streaming platform of a variety of content types, the objective of this work is to measure the quality degradation of a Skype video call under certain network conditions. The lack of documentation on Skype’s proprietary streaming protocol and the large use of obfuscation techniques over the transmitted data allows no control over the packet payloads and identification or alignment of packets and frames sent by a Skype client. Therefore, these constraints urged the development of a custom testbed to compare the delivery against the original video quality as perceived by the users. The testbed is developed in three phases:

(i) Video injection at the source side: the use of specific videos in raw format injected into Skype instead of camera output allows for controlled experiments and a plethora of content types.

(ii) Automated transmission of the input videos and impairment of the transmission channel: the channel is impaired with a commercial network emulator (PacketStorm Hurricane II) able to emulate a variety of network topologies and conditions.

(iii) Video recording and evaluation at the receiver side: the received video is captured in raw format at the receiver, synchronized and compared to the original frame sequence in terms of quality degradation.

\(^c\)http://www.packetstorm.com.
To ensure more accurate and unbiased experiments, we used a direct connection between the two clients and the network emulator isolated from any other traffic. However, we allowed communication of Skype clients to Skype login servers via additional network interfaces and different network. As shown in Fig. 1, the overall architecture consists of two hosts running Skype clients and the network emulator between them. The hosts are also connected to the Internet via a local router. In this way, source (A) and sink (B) are forced to communicate only through the emulator (C) that can emulate a variety of standardized and customized impairments in the network. These network conditions allowed Skype to operate with UDP protocol for streaming without using any other Skype supernode on the Internet.

We developed a Skype plugin to automatically select the video from file system, read user input for frame size and rate, and finally start a Skype video call to a specific user at sink B. The original Skype application can only accept input from a camera. As Skype has not released the video API of the application, we used e2eSoft VCam open SDK\textsuperscript{d} to fake the camera input. This was the best option to push a video file into the latest Windows version of Skype via Skype4COM API. Avoiding loss of information, input video was in raw RGB24 format read from a binary file and pushed to the VCam driver frame by frame at a certain rate. With the use of the raw video file, we eliminated any loss of information before the video frames had been handed over to Skype for streaming. Moreover, it enables direct pixel-by-pixel comparisons between original and recorded videos. Encoding the video before pushing it into VCam driver would make it impossible for Skype to apply its own encoding schemes.

Skype video recording was an issue and many available tools were tested. On Windows platform, there are many commercial and non-free applications able to

\textsuperscript{d}http://www.e2esoft.cn/vcam.
record the Skype video screen but none of them was robust enough to record at high rates like 25 fps or 30 fps and most of them did not allow raw format. Thus, in first attempt, we used Camtasia Studio\textsuperscript{e} to record a screen big enough to handle the resolution of the video and store it in raw format. This way of recording introduced a number of problems and, hence, all the experiments provided useful information about the validation of the testbed. We group those experiments together and label them as Phase 1. Typical problems raised in this phase were:

- Oversampling: most of the time Skype was unable to deliver 30 fps but Camtasia recorded 30 fps constantly.
- Regardless the network impairments, frames were delivered with a varying time lag. Hence, the recorded video was stretched in a non-uniform way.
- Many frames were lost especially at the beginning of the stream or when high packet loss was imposed in the stream.
- Some pixels of the received video with Windows Skype clients were cut in the display and this led to a stretch distortion in the recorded frames.

To tackle these problems, we moved to Phase 2 of experiments with a Linux-based sink. Using ffmpeg\textsuperscript{f} on the sink platform with the X11Grabbing plugin, we were able to directly record the screen from the graphic server. The Linux versions of Skype did not exhibit the “crop” issue; hence, we recorded exactly what was received at the sink. For the remaining problems, we developed a small application that tried to align the recorded frames with the transmitted frames taking into account all the distortions on the stream.\textsuperscript{g} We introduced frame indexing within the video stream. The idea behind this indexing technique is a black and white progress bar embedded in each frame which grows for every transmitted frame. The sink can calculate the frame number that was actually received or dropped. The width of each step of that progress bar (i.e. number of white pixels added to the previous step) has to be big enough to survive even the most severe impairments. However, the length of the bar might get much bigger than the width of the frame depending on the number of frames transmitted.

Let us assume a video with frame width $w$ (e.g. $w = 640$) pixels wide and a step of the bar equal to $b$ (e.g. $b = 5$) pixels wide. That is, a $w$-wide bar can accommodate $r = w/b$ (e.g. $r = 640/5 = 128$) frames. Every $r$ frames a new cycle starts. Therefore, we embedded a cyclic progress bar, which fills up from left to right with white pixels during odd rounds and with black pixels during the even rounds. Due to distortions explained above, many frames were repeated in the recorded video. The indexing technique with the progress bar helped us identify the distinct frames as well as the number of their repetitions. This index was enough to reconstruct the video with exactly the same number of frames as the original. In case of a gap, i.e. lost frames, the last displayed frame before the gap was used.

\textsuperscript{e}http://www.twchsmith.com/camtasia.html
\textsuperscript{f}http://ffmpeg.org
\textsuperscript{g}
The procedure above resulted in RGB24 videos recorded from Skype containing the impairments introduced by Skype codec and impairment box. The size of the videos was the same as their original; hence, structural similarity analysis could be applied.

4. Design of Experiments

In these experiments, we used different videos of the same length and resolution but very different in terms of motion and colors.

The first clip is a video file named “shields” picked from the Live Video Quality Database.\(^8\) The original file was a YUV420 video of 250 frames at 1920 × 1080 resolution. We have chosen a video with a mixture of high and low motions which allows to test both conditions at the same time. As shown in Fig. 2(a), the chosen video has long periods of high temporal information index and short periods of low motion. The clip was then repeated four times to produce a longer video clip (1000 frames) so that we analyzed both the transient and the steady phase of a Skype call. The “shields” video covered Phase 1 of tests which focused on the validation of the testbed and the effect of limited computational resources to the quality of the stream.

The second sequence is a “head&shoulder” clip (from now on simply named “h&s”) provided by Skype on its site for developers\(^h\) showing a girl talking over

\[\text{Fig. 2. Temporal information for the (a) “shields”, (b) “h&s” and (c) “birds” clips.}\]

\(^8\)http://live.ece.utexas.edu/research/quality/live_video.html.
\(^h\)developer.skype.com.
a fixed background. This video was chosen because of the hypothesis that Skype video engine could perform better with the common static video call content that is usually transmitted. For that reason, we used it as a “benchmark” to evaluate the results for the favorable and unfavorable cases. The video has low motions as shown by temporal information index in Fig. 2(b).

The third video is a more complex clip picked from The Consumer Digital Video Library, showing several scenes with lots of motions, about a girl sat on a bench and birds flying away from a lake (from now on simply named “birds”). One peculiarity of this video is the presence of two consecutive equal frames at every five of them. That peculiarity is reflected by the temporal information index, as shown in Fig. 2(c), which presents high variability with periodical drops.

The last two videos covered Phase 2 with focus on the assessment of video quality under different network conditions and types of impairments. These were short clips of 300 frames at 1920 × 1080 resolution. For these clips, we concatenated the original clip eight times (i.e. 2400 frames per clip) to secure more accurate results and to allow Skype to stabilize after the call initiation phase. Despite the time-constraint operation of video conferencing, Skype uses a limited buffer and network probes to estimate the network condition and optimize the delivery when a call is opened. The duration of this initiation phase can vary depending on multiple parameters, such as network conditions and computation resources at the hosting environments. However, the resolution was too high for transmitting through Skype because of the huge processing power required. Therefore, we decided to downscale the videos to 640 × 320 pixels maintaining the aspect ratio and converting the videos to RGB24 format for the virtual camera (e2esoft vcam). This resolution is what Skype calls “High Quality”.

The Skype plug-in we developed on the sender side was reading the video file frame-by-frame adding the appropriate step of the progress bar on top of the video. Hence, the resulting clips really transmitted by Skype were at 640 × 480 pixels. Every frame was then pushed to the VCam driver at the correct frame rate (25 fps or 30 fps depending on the video). An example of that frame indexing scheme and the overall video assessment testbed is presented in Fig. 3. Figure 3(a) illustrates four frames from the transmitted video. The video alignment process at the sink extracts the top 160px, increases the contrast and measures the bar length. Assuming the frame is at cycle $K$, if $K$ is an odd number then the progress bar is white; otherwise, the bar is black. For instance, the white bars in Fig. 3(a) have lengths $i$ and $j$ respectively, and represent the $i$th and $j$th frames of the $K$th cycle. Symmetrically, the black bars represent the $m$th and $n$th frames of the $(K + 1)$th cycle.

Both source and sink were running on an Intel Core 2 Quad with 4GB DDR3 memory and Intel Q45 Express Graphic cards. Skype version on the source (Windows 7 OS), it was 5.5.59.124 while on the sink (Ubuntu Linux OS), it was the 2.2.0.35 beta. The network emulator used in these experiments is the PacketStorm

Hurricane II. Hurricane II has multiple ports for both fast Ethernet, and Gigabit Ethernet, and can provide network conditions in a repeatable and controllable setting. It is also capable of network capturing and real-time statistics. For our target, we connected the PacketStorm to the client machines and impaired only the path that went from the source to the sink. Neither the feedback path (signaling from sink to source) nor the connection to Skype login servers was conditioned.

We defined three kinds of experiments based on different impairment configurations set up on the network emulator:

(i) No impairments imposed on the stream; that is, no packets were dropped by the delivery channel.

(ii) Gaussian random packet drop rate and/or bursts of dropped packets. It was by choice to leave the feedback channel intact to make sure that our target (analysis of Skype video reconstruction mechanism) was as unbiased as possible. Packets of the stream were dropped in two ways. The first part of experiments imposed a random gaussian packet drop distribution with values between 0–20% and mean value equal to 10%. That is, every second of stream 0–20% and on average 10% of the packets sent by the source were dropped. Besides that packet drop rate, the second part of experiments enforces packet drop in bursts. That is, for every dropped packet the next 4 are also dropped (length of burst equals to 5) but the total percentage of dropped packets per second was following the gaussian distribution described before. In the third part, we modified the distribution to dropped packets to the average of 2% (min: 0%, max: 4%) with the same burst length of 5 to investigate how changes in these parameters affected the quality.

(iii) Uniform distributed jitter of 75 ms in the network.
At the source side, we set up a virtual machine running IPCop Linux operating system in which we have configured a small router using IPTables to enable the source to be connected to both the Internet and network emulator. At the sink, the X11 grabbing plugin of ffmpeg, was recording the screen at desired rate without any compression in RGB24 format. The screen resolution was kept at $640 \times 480$ and the recording was made fullscreen to avoid any potentially loss of pixel within the recording area. The recorded video was trimmed to start and finish exactly at the first and last displayed frames. The resulting file was then split in two clips: one containing only the progress bar ($640 \times 160$ pixels) and the other with the real video content ($640 \times 320$ pixels). The index was calculated, processing the progress bar video which was used to filter the recorded content and to produce the final content video. To reduce the probability of errors while calculating the index of each frame, we maximized the contrast of progress bar frames. Finally, we used a small Java application$^{10}$ to compute SSIM values of the final recorded video compared to the original one. Figure 3(b) gives a schematic description of this procedure.

5. Results and Discussions

Running the experiments above produced a number of results presented in this section. The results are classified in three areas: first, we validate the testbed; second, we assess Skype video quality streamed over channels impaired with packet loss and; and third, we impaired with jitter.

5.1. Phase 1: Video quality under limited computational resources

In this section, we identify possible reasons for quality degradation as an important step to optimize the testbed and reduce the bias to our measurements. Aligning the recorded and original videos revealed that many frames were never displayed at the sink, even on the experiment without impairments. This reflects the inability of Skype player to reconstruct the dropped frames, either because its streaming counterpart never sent them or there was not enough information or computation power available on time to build the frames.

As Fig. 4(a) illustrates, more than 400 out of 1000 frames at “no-loss” and “low loss” experiments were never displayed. “High loss” experiments double the packet drop rate of “low loss” experiments and the number of dropped frames. The number of frames displayed exactly once is almost the same for the two “no-loss” and “low loss” experiments with a slight advantage of the first. However, for “high loss” experiments show a dis-proportionally low number of frames displayed exactly once and some frames displayed multiple times. The frames displayed twice or more times practically covered gaps left by dropped frames. This is a clear indication that the testbed is able to capture the differences in quality imposed on the video stream by network conditions.

$^{10}$
Fig. 4. (a) Histogram showing the number of frames that were dropped or displayed once, twice or more times, respectively, on all the three experiments. (b) SSIM values for “shields” video without network impairments.

Fig. 4(b) illustrates the SSIM index values for the “shield” video as moving average of 25 frames. Each cycle is one repetition of the original video clip (250 frames). When the video shifts from one cycle to the next, the SSIM index decreases. The temporal difference between the last frame of a cycle and the first ones of the next cycle is high. This forces Skype to send more data. Since the aligning software puts the last received available frame in place of dropped ones, when a high packet loss occurs, the SSIM index decreases especially during the transition between two cycles. This is clear because the difference between the scenes at the beginning and at the end of the videos is typically high.

Analyzing the packets transmitted during the “no-loss” experiment, we can assert that no packets were lost traversing the network. In that case, the impairments come from either the communication end-points (e.g. computation resources at source and sink are not enough to handle the frame rate and size) or Skype protocol. However, this assumption is not secure as there was no way to confirm through the payload of captured packets the frames that were actually transmitted and dropped in the source side and the ones that were rendered in the receiver side.
The objective quality of the video streamed over “high loss” channel, Fig. 5(b), has degraded much more than over “low loss” channel, Fig. 5(a). Comparing Fig. 5(a) with Fig. 4(b), it becomes clear that despite the lossy channel, the video quality has not been significantly degraded. This is to confirm that some of the frames might be dropped for reasons other than the network impairments. If Skype protocol does not cause those drops, computation resources might introduce a bias to our measurements. In Phase 2, we readjust our measurements to reduce that potential bias.

For every dropped frame, the last available one is displayed. Hence, the more consecutive frames are dropped, the lower the SSIM index gets. In the last part of the simulation we can see an improvement of SSIM and based on the loss pattern, this cannot be the result of lower drop rate. In fact there are also similar values during higher drop periods, thus this behavior can only be related to the video content. This could be an indication that Skype assumes low motion videos such as typical “h&s” video content.

Further experiments with other video types, i.e. “h&s” and “birds”, clarify the assumptions above. The no-loss SSIM index in Fig. 6 depicts a peculiarity: the “h&s” results present a very high and almost stable trend while “birds” have a variable behavior repeated over each cycle of the original video clip (300 frames in this case). Even when the stream is not impaired, Skype still suffers from high motion videos resulting in quality degradation. This problem is even more evident in the experiment with the “birds” video in which the motion is higher and the difference between the first and the last frames is bigger. Therefore, the asymmetry in the contents in terms of temporal and spatial information affects the quality of the received video.

Comparing the temporal information values in Fig. 2 with the results shown in Figs. 4(b) and 6, there is a mismatch between the motion level of the videos and their corresponding quality degradation in experiments without impairments. “Shields” video has higher values of temporal information than “birds” but both videos have periods with temporal information index between 8 and 10.
the SSIM values of the two videos are quite different and “birds” video seems to perform better. That contradicts with the initial hypothesis that Skype is optimized for typical static video call communication. Since there is no way to detect where the problem came from, we improve the available computational power and recording reliability. “Shields” experiments have used Camtasia Studio for recording the delivered video. For that reason in this work we moved to ffmpeg and Linux platform at the sink trying to improve the reliability of our experiments.

5.2. Phase 2: Skype stream stressed with packet loss

Using the results of Phase 1, we implemented the appropriate improvements to our testbed and conducted experiments over impaired channels as described in Sec. 4. Figure 7 illustrates the SSIM values for the two videos when the normal packet drop distribution with mean = 10% is affecting the stream. There are long periods in “h&s” experiment in which the SSIM index value is very close to the one calculated without impairments. This confirms the hypothesis that Skype treats better “head&shoulder” videos. Indeed, the “birds” results present a bigger deviation from the no-impairment trend line confirming the assumption.

In Fig. 8, the plots show the quality degradation of the two videos (“h&s” and “birds”) in case of burst packet drop rate. The normal packet drop distribution explained above is applied in bursts of 5 consecutive packets. The quality in “h&s” video decreases after 20 seconds of emulation following a big drop event. From that point onwards, SSIM value stays around 90% until the end except for the packet loss peaks which also further but temporarily deteriorate the quality. On the contrary, the results for “birds” video in Fig. 7(b) present a pattern after 30 seconds while
Fig. 7. SSIM values for (a) “h&s” and (b) “birds” videos with normal packet drop distribution around mean value of 10%.

Fig. 8. SSIM values for (a) “h&s” and (b) “birds” videos with 10% random packet drop impairments and burst length of 5.

in Fig. 8(b) this pattern begins already after 10 seconds. The quality for this video on the randomly lossy channel (Fig. 7) is higher at the beginning but much slower to stabilize compared to the burst lossy channel of Fig. 8.

The SSIM results are very similar to the previous simulation even if the total number of dropped packets is higher. This is a rather unexpected result because Skype reaches the same quality even when the receiver client has less information to reconstruct the frames. To go deeper in this analysis, we tried to decrease the packet drop rate to 2%, maintaining the burst length at 5 and reducing the frequency of drop events. With these conditions as shown in Fig. 9, the quality rises again and stays high for the whole call. This means that bursts of dropped packets affect the Skype video quality to a lesser extent. For randomly lost packets however, Skype encounters problems on reconstructing the video and the quality drops.

5.3. Phase 2: Video quality degradation under jitter

We perform a last experiment impairing the stream with jitter as shown in Fig. 10. We set up the emulator with a uniformly distributed value of 75 ms. For “h&s”,

G. Exarchakos et al.
video quality oscillates around 90% from the beginning until the end of the emulation following always the same pattern as that in the drop experiments but a little bit worse in SSIM values. For “birds” video in Fig. 10(b) there is no pattern anymore but a significant quality degradation with the quality showing an upward trend at the second half of the emulation. Though the degradation in case of high motion videos can be significant, Skype continuously probes the network connection and it seems to be able to gradually recover video quality from delayed packets better than losses.

6. Conclusion

In this paper we conduct an objective QoE analysis of Skype video streams using a variety of video content. Our target was to find the correlations between the objective video quality and the underlying network conditions. We streamed specific raw videos through a Skype plug-in and then impaired the channel with the help of a commercial network emulator. The content was recorded as seen on the screen
and aligned with the original sequence. This allowed us to compare original and received video and calculate SSIM index values.

The results gave us certain useful conclusions. When there are no losses in the network, Skype is able to reconstruct most of the frames in that interval in any type of video. Video content and its temporal information seem to play an important role in the delivered quality even in case of unimpaired communication channel. Our experiments have shown a weakness of Skype to handle the variation of temporal information and, at lesser extent, high temporal information. Very low temporal information index (see “h&s” in Fig. 2(b)) can be handled with ease in both unimpaired and impaired scenarios. On the contrary, high motion videos that exhibit relatively smooth temporal information index throughout the video (see “shields” in Fig. 2(a)) forces a significant quality degradation. Videos with temporal information index in between “h&s” and “shields” but with high variability can decrease the delivered quality even further.

Skype streams are affected by packet loss as Skype has not enough information to reconstruct the frames and, therefore, is forced to drop them. Lossy channels stress more evidently the weakness of Skype to handle high motion videos and its advantage in dealing with low motion ones. Moreover, Skype shows weakness to recover fast from a high and long packet loss. We introduced drop impairments in the network and discovered that Skype behaves in different ways depending on the packet loss distributions. We showed that Skype adapts video quality based just on network packet loss rate; that is, it is not severely affected by bursts of lost packets. We extended the experiments with jitter impairments and showed the differences from the previous test results. High motion videos seem to experience significant quality degradation from jitter but Skype can recover the quality faster than that from packet losses, exhibiting a relative resilience to delay as opposed to packet loss.

Ongoing work and future efforts focus further and more detailed experiments that correlate a variety of network impairments with Skype video quality. We are working towards a fully automatic and real-time video quality estimation mechanism more compliant to QoE than QoS. In this regard we try to develop a better video alignment process as the progress bar used for this work might introduce some doubts on the reliability of the results. Finally, within our future plans, we will also assess Skype video quality with subjective methods and Adaptive MLDS.11

References


1350029-16


