EVALUATING NETWORK PERFORMANCE IN MOBILE MULTIMEDIA DELIVERY

Srdjan Sladojevic[1], Dubravko Culibrk[1], Milan Mirkovic[1], Marko Panic[1]

[1]*Fakultet tehnickih nauka, Novi Sad*

Abstract – *The paper proposes a framework for network performance evaluation in multimedia delivery to mobile devices. The proposed approach is based on logging the sequences of received packets during the transmission of H.264 coded video using the common Real Time Streaming Protocol (RTSP). Logged sequences enable the monitoring of the Quality of Service (QoS) provided to the user and subsequent reconstruction of impaired video sequences as they were presented to the user. We evaluated the proposed approach in three different, realistic multimedia delivery scenarios. The effects of such transmission conditions are explored based on a set of videos typically used to evaluate multimedia system performance. We further evaluate the system's performance based on the number of successfully decoded frames.*

1. INTRODUCTION

With the rapid increase in wireless video transfer to mobile devices, it is important to monitor the quality of received video. During the transfer it passes through distortion channels in error-prone networks where quality is reduced through operations such as compression, channel coding, transmission errors, error concealment mechanisms, decoding, etc.

In an overwhelming majority of applications, the end-user of a video sequence is a human observer [1]. It is hence of interest to evaluate the quality and performance of a video transmission and delivery system, as it is perceived by a human observer.

To simulate the effect of transmission conditions on the transmitted multimedia content, researchers typically use channel simulator software. In this paper we propose a different approach based on recording transmission errors that occur in real-world transmission scenarios. To achieve this, we capture the traces (log the information) pertinent to the packets received during the streaming of a video via a wireless network. In addition to providing valuable information about the performance of the network from the multimedia delivery standpoint, these logs can be provided as input to a PC application, that enables one to generate the impaired video as it would be acquired by the mobile device. There are several advantages of this approach over simply capturing the decoded frames of the streamed video on a mobile device: (i) capturing videos can be very time consuming, (ii) it can violate packet receiving schedule and (iii) it is possible to introduce impairments pertinent to different network conditions to new videos and thus build a test-set of arbitrary size relatively easy. For the purpose of this paper, we used the developed system to record packet reception sequences during different transmission scenarios in three typical use-cases. We analyze the results obtained and provide a selection of interesting patterns and that can be downloaded from [2] and used freely for research purposes.

The rest of the paper is organized as follows: Section 2 provides an overview of the related work in the field. Section 3 describes the proposed approach in more detail. Section 4 deals with the experiments conducted and results achieved. Section 5 holds our conclusions.

2. RELATED WORK

The performance of multimedia transmission systems is usually evaluated either from the standpoint of visual quality of digital videos as perceived by human observers or based on standard network QoS measures.

There is not a lot of work concerned with creating realistic traces of multimedia transmission over the wireless and network performance. Fitzek et al. [3] describe an early study (2001) that was aimed at generating frame size traces (logs) for videos in QCIF resolution, coded using MPEG-4 and, now largely obsolete, H.263. Seeling et al. [4] extended the study to layered video, basing the performance evaluation on frame size and quality traces. The quality was represented by PSNR, although it is widely known to be a poor representative of quality.

Although network-related errors are rarely considered in studies of network performance, they are of interest in different studies aimed at Video Quality Assessment (VQA) [1]. Basic data for VQA research takes the form of databases of videos which contain simulated impairments. LIVE (Laboratory for Image and Video Engineering) database, provided by authors of [5], is one of the most commonly used databases when VQA-related tasks are in question. The database contains four types of distorted test videos: MPEG-2 compressed, H.264 compressed, Wireless and IP distorted videos. Network effects were simulated as IP losses on an H.264 compressed video stream. Error patterns were obtained from real-world experiments on

congested wired networks and are recommended by the Video Coding Experts Group (VCEG) to simulate Internet backbone performance for video coding experiments [6][7]. Seshadrinathan et al. [5] simulated errors in wireless environments using bit error patterns and software available from the VCEG [9].

In the IVP database [10], the packet loss that occurs when transmitting H.264 videos was simulated at a rate of 0.1%, 0.5%, 1%, 3% or 5%, using the error patterns recommended by the VCEG to simulate the Internet backbone performance for video coding experiments [11]. The patterns themselves are a result of early real-world experiments on Internet backbone.

To the best of our knowledge, there are no studies attempting to evaluate network performance by logging the RTP packet reception directly on a mobile device, nor is there a publicly available framework that allows for impairment to be introduced to videos, based on logged traces. Both issues are addressed by the framework proposed here.

3. PROPOSED APPROACH

The architecture of the proposed system is shown in Figure 1. The video is streamed from a server to a mobile device, via a wireless network. LIVE555 Media Server was used to stream H.264 media. It is part of LIVE555 Streaming Media, which is a set of open source (LGPL) C++ libraries for multimedia streaming. The libraries support open standards such as RTP/RTCP and RTSP for streaming, and can also manage video formats such as H.264, MPEG, VP8, and DV, and audio formats such as MPEG, AMR, AC-3 and Vorbis [12]. Media Server is a complete RTSP server application and by default, the server transmits its streams as RTP/UDP packets. We used it in the default mode. The server was deployed on a Linux box.

The original server source code had to be changed to enable packet reception logging. Every RTP packet contains sequence number in its header data. Originally, the first number in the sequence was a random number generated by the server. This had to be changed to start from 1 in each streaming session to enable accurate reproduction of the same streaming conditions later.

The key component of the proposed system is an Android RTSP client-side application that had to be developed. It uses the FFMpeg library for streaming [13]. FFMpeg code was altered in order to be able to record all received packets' sequence numbers to a log file, along with the timestamp of reception of each packet as well as the timestamp when each packet was decoded.

Figure 1. Architecture of the system used to record received packets

Once recorded, the logs can be used to reproduce the network conditions during the original transmission. To enable the visual inspection of the effect of different transmission conditions on the video transmitted, as well as the creation of impaired videos for the purposes of VQA, a video impairment application has been developed. Running on a PC it is able to receive the video stream from the same stream server, as in the original transmission. Using a log file as input, the application drops packets reported missing by the log file, and decodes video frames using the rest of the packets. Timestamps from the log file are used to maintain the dynamics of the original transmission and allow the system to feed the packets to the decoder with the delay experienced during the original transmission. A block diagram of the video impairment process is shown in Figure 2.

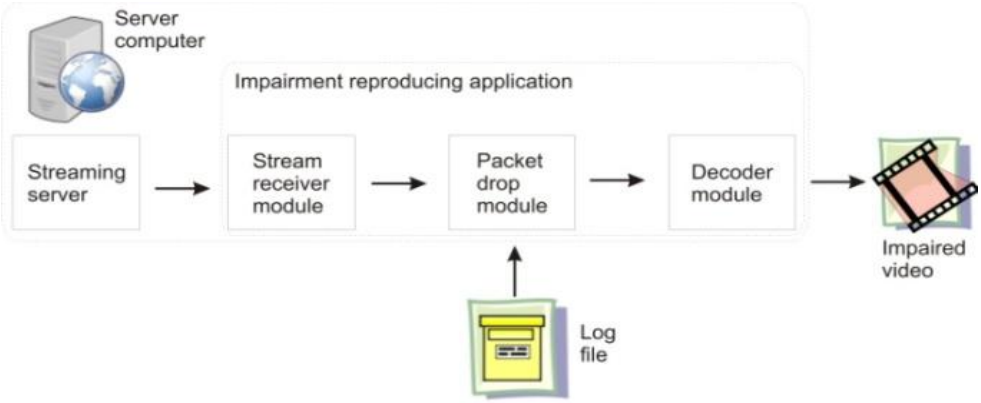

Figure 2. Impairment generation process.

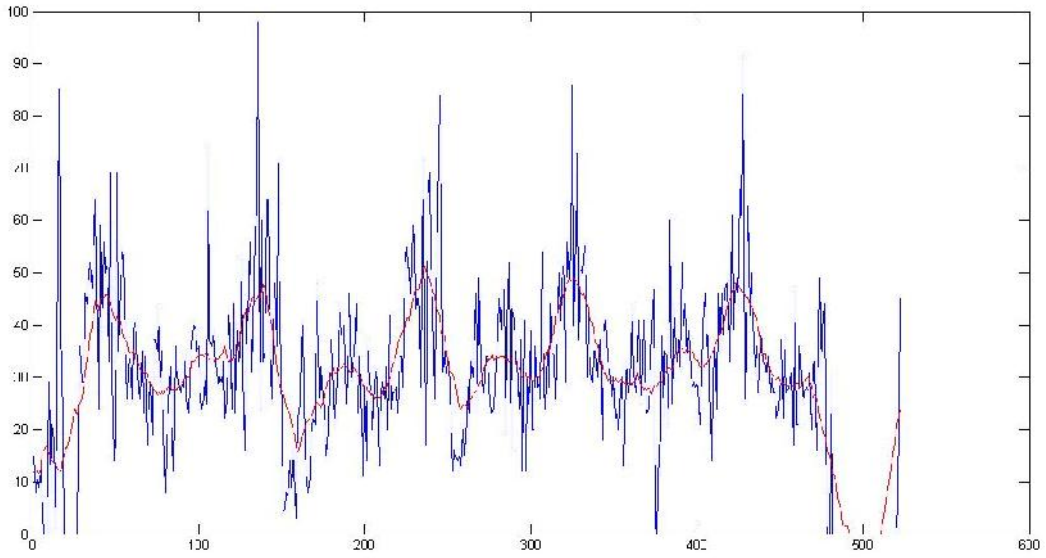

Figure 3. Number of received RTP packets per second.

4. EXPERIMENTS AND RESULTS

To be able to measure network performance, one first needs to select the media content to be streamed. For the purpose of experiments described here, we used standard videos available within the LIVE Video Quality Database [5]. This is a common source of content for comparative evaluation of newer objective quality assessment models and state-of-the-art in perceptual quality evaluation systems. A subset of videos available in the database was selected and it includes the following videos: Pedestrian Area, River Bed, Rush Hour, Tractor, Station, Sunflower, Blue Sky, Shield, Park Run and Mobile & Calendar.

Original video files were in planar YUV 4:2:0 format and did not contain any headers. The spatial resolution of all videos was 768x432 pixels. VirtualDub [14] was used to resize these videos to resolution of 384x216, as this resolution is more appropriate for playback on mobile devices. In addition, all videos that had a frame rate of 50 fps were converted to 25 fps. Once again, this is more suitable for mobile devices. Finally, they were compressed using x264vfw VirtualDub plug-in. x264vfw is the VfW (Video for Windows) version of a well known x264 encoder + ffh264 decoder (from FFmpeg/Libav project).

Finally, all videos were concatenated to form a single sequence that was streamed in a single session. The sequence is 493s long, and has 12325 frames. This video was split into 29615 RTP packets.

The experiments were conducted in Florida, USA in July and August in 2012. The equipment used was as follows: the streaming server ran on a Core2DUO PC 2.5 GHz, with the Ubuntu 12.04 operating system. The mobile device used to receive streams was a HTC Desire mobile phone with an Android 2.2 operating system. The network provider was AT&T. The mobile device was receiving the content over the GSM network and the Florida Atlantic University WiFi network, for the last scenario.

We explored three common scenarios, when video streaming on mobile devices is in question: (i) static scenario, where mobile device was motionless, (ii) driving scenario, where streaming was performed while driving on the highway at the average speed of around 65 mph, and (iii) WiFi scenario, where content was streamed over WiFi network to a motionless mobile device.

Figure 3 shows the plot of the number of packets received per second for the WiFi scenario. The running average of the number of packets over an interval of 20 seconds is indicated in red. As the figure shows, even in the case where the device is static and video streamed over WiFi, there are significant fluctuations in the number of packets received over time.

The overall number of packets received and frames decoded in different scenarios was: (i) 9210 packets received and 4377 frames decoded for the static scenario. (ii) 1908 packets received and 657 frames decoded for the highway scenario. (iii) 11837 packets received and 5239 frames decoded for the WiFi scenario.

(a) Reference

(b) Static device over GSM impaired frame (best segment).

(c) Reference

(d) Static device over GSM impaired frame (worst segment)

(e) Reference

(f) Driving scenario impaired frame.

Figure 4. Example of impaired videos.

To examine the impact of different network conditions on the quality of video received by the device, we first needed to select short subsets of the traces similar to that shown in Figure 3, as the individual sequences are shorter than our traces. The plots were analyzed using Matlab and the following 20-second excerpts were extracted from the logs: the best(Sc1) and worst (Sc2) segment for the first scenario, the best (Sc3) and average (Sc4) segment for the highway scenario and the best segment for the WiFi (Sc5) scenario. The segments selected were then used to impair videos from our database.

The impairment introduction application was run on the same PC as the server, in order to avoid additional packet loss during simulated transmission. Thus, both the client and the server ran on the same computer. The decoded frames were saved to disk as bitmaps.

Finally the decoded frames were merged to form impaired sequences using Matlab. The script used adds decoded frames into appropriate order, taking into account the decoded frame timestamps, as they are provided by the decoder. If frames are missing, we substitute them with a copy of the last decoded frame, creating the “freeze” effect.

Figure 4 shows sample frames from impaired sequences. The impaired frame shown in Figure 4 (b) is a frame from a sequence degraded using Sc1. Artefacts shown in Figure 4 (d) are caused in the worst segment of the static device receiving over GSM (Sc2), while the artefacts shown in Figure 4 (f) were generated in the driving scenario (Sc4).

For each reference video, 5 corresponding impaired videos were created, one for each scenario e.g. for video bs, videos bs1, bs2, bs3, bs4 and bs5 were

created and each of them represents artefacts introduced by transmission conditions Sc1, Sc2, Sc3, Sc4 and Sc5. To evaluate the performance of the network in these cases we examined the number of frames decoded in each scenario. These values are shown in Table 1, for different videos. As the table shows, 61.2% of frames were decoded in scenario Sc1, which represents the best performance, followed closely by scenario Sc5, where 60.4% of frames were successfully decoded. The performance in the worst segment of the static GSM delivery (Sc2) is significantly lower than that achieved in the best segment of the same transmission scenario and only slightly better than that achieved in the best section of the highway streaming (Sc3). In Sc2 17.3% of frames could be decoded while in Sc3, 15.6% of frames could be decoded. The worst results were achieved in scenario Sc4 where only 6.6% of frames could be decoded. This is the reason why the worst segment of the highway scenario was not considered, as there no meaningful results could be obtained in that case.

5. CONCLUSION

A framework for network performance evaluation in multimedia delivery to mobile devices has been proposed. The approach is based on logging the sequences of received packets during the transmission of H.264 coded video using the commonly used RTSP protocol. We record the sequence numbers and timestamps of individual packets on a mobile device to create a trace (log) of the transmission. The logs thus obtained can be used to analyze network performance in specific scenarios and impair new videos using a PC application that has been developed within the study.

In the paper we present the results of evaluating the network performance in the common mobile device use scenarios: static device over GSM and WiFi and viewing a video while in a moving vehicle over GSM. As expected, our results indicate that the best delivery, both in terms of packets and decoded frames is achieved when the device is static. The worst segments in our static GSM trace are very close to those obtained in the best segments of the moving vehicle scenario, while the average performance of streaming videos to a device in a moving vehicle over GSM is very poor and only 6% of the frames are successfully decoded.

Recorded RTP packet reception (loss) patterns can be used create realistic impairments for the purposes of VQA research and have been made public. The framework itself can be used by the service providers as a diagnostic device, when wireless multimedia transmission is concerned.

Table 1. Number of frames decoded in each scenario

	Video (yuv)	Frm No	Frames decoded in scenario				
			Sc 1	Sc 2	Sc 3	Sc4	Sc 5
1	bs	217	137	35	50	16	127
2	mc	250	172	26	35	13	137
3	pa	250	164	54	40	17	166
4	pr	250	124	57	44	12	144
5	rb	250	78	24	39	17	128
6	rh	250	169	55	33	21	167
7	sf	250	168	53	23	17	162
8	sh	250	169	42	42	18	153
9	st	250	170	44	33	19	166
10	tr	250	158	36	47	13	140
Frames decoded (%)			61.2	17.3	15.6	6.6	60.4

ACKNOWLEDGEMENT

This research was supported by the FP7 Marie-Curie project QoSTREAM (Grant Agreement 295220) and the Serbian Ministry for Education, Science and Technology Development under grants III43002 and III44003.

REFERENCES

[1] Seshadrinathan, Kalpana, and Alan C. Bovik. "An information theoretic video quality metric based on motion models." Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics. 2007.

[2] http://www.vidqual.com/vqa/RTPPacketLossPatterns.zip

[3] Fitzek, Frank HP, and Martin Reisslein. "MPEG-4 and H. 263 video traces for network performance evaluation." Network, IEEE 15, no. 6 (2001): 40-54.

[4] Seeling, Patrick, Martin Reisslein, and Beshan Kulapala. "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial." Communications Surveys & Tutorials, IEEE 6, no. 3 (2004): 58-78.

[5] K. Seshadrinathan, R. Soundararajan, A.C. Bovik, L.K. Cormack, Study of Subjective and Objective Quality Assessment of Video, IEEE Transactions on Image Processing, vol. 19, no.6, pp. 1427–1441, June 2010.

[6] (2005) International organization for standardization. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c039486 ISO IEC 13818-5 2005 Reference Software.zip

[7] “H.264/AVC software coordination”. Internet: http://iphome.hhi.de/suehring/tml/, [Dec. 06, 2012]
[8] “Proposed error patterns for Internet experiments”. Internet: http://wftp3.itu.int/av-arch/video-site/9910_Red/q15i16.zip, [Dec. 06, 2012]
[9] (1999) Common test conditions for RTP/IP over 3GPP/3GPP2. Internet: http://ftp3.itu.ch/av-arch/video-site/0109 San/VCEG-N80software.zip
[10] F. Zhang, S. Li, L. Ma, Y. C. Wong, and K. N. Ngan, “IVP subjective quality video database,” 2011, Internet: http://ivp.ee.cuhk.edu.hk/research/database/subjective/.
[11] “Proposed error patterns for Internet experiments”. Internet: http://wftp3.itu.int/av-arch/video-site/9910_Red/q15i16.zip, [Dec. 06, 2012]
[12] Live Networks, Inc. “The LIVE555 Media Server”. Internet: http://www.live555.com/mediaServer/#about, [Dec. 06, 2012]
[13] F. Bellard, M. Niedermayer, et al., 2007, FFmpeg. Internet: http://www.ffmpeg.org, [Dec. 27, 2012]
[14] VirtualDub software. Internet: http://www.virtualdub.org/, [Jan. 23, 2012]