StreamSim: A Video Streaming Simulation Toolchain for Unreliable Transport Mechanisms

Alexander M. Dethof*, Werner Robitza†, Marie-Neige Garcia*
*Telekom Innovation Laboratories, Assessment of IP-based Applications, TU Berlin, Germany
†Telekom Innovation Laboratories, Deutsche Telekom AG, Berlin, Germany

***Abstract*—This paper proposes a highly flexible framework to stream videos using unreliable transport mechanisms involving combinations of MPEG-2 TS, RTP and UDP. The framework currently supports the H.264 and H.265 codecs. Transmission impairments such as packet loss, delay and jitter can also be simulated during streaming or offline. Various adjustable models of impairment distributions such as the Gilbert-Elliot model are implemented. By updating its libraries, the framework can easily be configured to use other codecs and transmission settings.**

I. Introduction

The new HEVC (H.265, MPEG-H Part 2) video codec supports up to Ultra High Definition (UHD) video resolutions and achieves better video quality than previous codecs such as H.264 / MPEG-4 AVC, especially for UHD videos [1].

Several studies have been conducted in order to investigate HEVC's robustness for different error concealment strategies in unreliable transport scenarios, which for instance yield packet loss [2]–[4]. These studies generally focus on NAL unit-based loss insertion [5]. However, with IP-based services like IPTV, transmission degradations such as packet loss occur at IP-packet level. To cover this more realistic case, we present a highly flexible and individually configurable framework, named *Video streaming simulation toolchain* (*StreamSim*), which can generate video streams with various corruptions based on online and offline manipulations, such as delay, jitter or packet loss inserted at IP-packet level. The framework is made freely and publicly available to facilitate reproducible research, and can be found on https://gitlab.com/vqeg/streamsim.

With its object-oriented approach, the framework has the ability to encode videos for different codecs – such as H.264 and H.265 – and to send the encoded stream over various network environments using different encapsulation techniques, such as MPEG-TS over UDP, optionally extended with RTP, or RTP only. If required, packet loss can be inserted offline without streaming through a pre-configured environment using a packet loss simulator. Packet loss traces can be generated with both online and offline loss insertion, therefore enabling the re-use of the trace with other streaming settings. The streamed video can then be decoded again.

II. Related Work

In [5], the *HEVStream* framework is proposed in order to test and evaluate the visual quality of streamed HEVC contents in different network environments. While our *StreamSim* aims at providing the results of each individual processing step for arbitrary encoding variants, the *HEVStream* framework uses separate coders with non-interleaved Single-Time Aggregation Packet (STAP-A) packetization on RTP/UDP transmission, as described in [6]. The streamer unit is implemented for two different network types: multi-homed mobile networks and single hybrid wired/wireless networks. For both network types the packets are scheduled for the transmission over given emulation routers with prior knowledge about the current network situation and by configuring different network constraints such as random packet loss rate and delay. Furthermore the decoder supports only a simple copy error concealment method for missing NAL units. No publicly available version of the *HEVStream* framework could be found.

Sirannon[1] is a tool with similar streaming and network simulation features as our proposed software. However it has not been updated for several years – thus, does not support HEVC – and cannot be easily compiled on current operating systems.

Finally, the *Telchemy Packet Capture (PCAP) loss insertion tool*[2] – which is also used in this framework – allows to apply packet loss, but it does not cover the entire encoding/streaming/decoding chain.

III. The Toolchain of StreamSim

StreamSim is a framework which makes use of a toolchain to simulate video streaming for various codecs, encoder and decoder implementations, network configurations and transmission protocols. The framework is primarily targeting HEVC and the influence of encoding and network settings for this codec. However, as previously mentioned, the toolchain can easily be used with other codecs.

As shown in Fig. 1, *StreamSim* uses five dedicated processing steps: "Encoding", "Streaming", "Manipulation", "Extraction" and "Decoding". Each step can be individually configured and thus behaves differently depending on the predeclared settings, which are defined in tree-like configuration tables. Intermediate files are stored at each processing step. The individual processing steps are further described in the following subsections.

Video sources (SRCs) are stored in AVI containers. Test conditions (Hypothetical Reference Circuits, HRCs) describe various encoding and transmission settings to be applied to the SRCs. The Processed Video Sequences (PVSes) correspond to the different combinations of SRCs and HRCs. SRCs, HRCs, PVS IDs and their characteristics are stored in separate configuration files.

[1] http://xstreamer.atlantis.ugent.be/
[2] http://vqegstl.ugent.be/?q=node/27

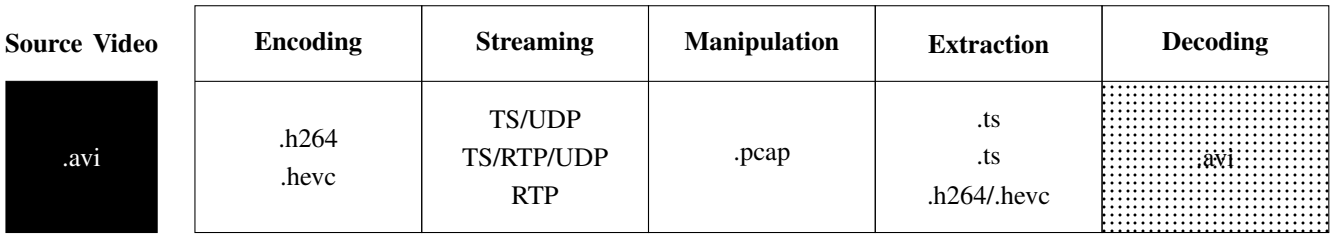

Fig. 1. *StreamSim* processing steps and related file formats.

1) Encoding: As a first step – "Encoding" in Fig. 1 – *StreamSim* encodes the SRCs according to the HRC descriptions. This step is performed for each SRC–HRC combination specified in the PVS configuration file. Regarding the encoding process only, an HRC contains information about the encoding software to use and the linked encoder settings. In *StreamSim*, *ffmpeg*[3] has been implemented as default coding software, featuring several popular encoders such as *libx264* and *libx265*. The encoded files are then stored in raw elementary stream files such as ".264" for H.264 and ".hevc" for H.265.

2) Streaming: In the basic streaming step – "Streaming" in Fig. 1 – the previously encoded videos are streamed by default to the localhost. If required, any other server can be used as receiver. The used protocol stack can be individually configured for each HRC setting. The following three variants are currently supported by the proposed framework: 1) MPEG-TS/UDP/IP, 2) MPEG-TS/RTP/UDP/IP and 3) RTP/IP. The network stream is then captured with *tcpdump*[4]. The packet dump is used as input for the network manipulation processing step.

3) Manipulation: In the third processing step – "Manipulation" in Fig. 1 – the packet stream can be manipulated either offline or online. In the offline case, the previously mentioned Telchemy packet loss insertion tool is used. The online manipulation is performed by re-streaming the previous packet stream to the same destination through a pre-configured network environment using the *TC/NETEM*[5] tool. For each manipulation type, various loss distribution patterns can be applied using the well-known Gilbert-Elliot and Markov-Chain models or any random configuration. Additionally, different delay distributions are provided by *TC/NETEM*. Finally, in the offline mode, loss traces in the form of text files can be applied on the PCAP files. Loss traces can also be captured in all cases by comparing the manipulated packet stream with the one before manipulation.

4) Extraction: The payload is extracted from the manipulated packet using *Scapy*[6]. If MPEG-2 TS encapsulation has been used during the "Streaming" process, the extracted payload is stored in MPEG-2 TS format. When no transport stream is applied, the extracted payload is stored as a raw elementary stream (e.g., ".h264" or ".hevc").

[3]http://www.ffmpeg.org
[4]http://www.tcpdump.org
[5]http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html
[6]http://www.secdev.org/projects/scapy

5) Decoding: In the "Decoding" step, the extracted payload is decoded back into an AVI container, using *ffmpeg* and the same codec as in the "Encoding" step. These decoded files can be used to compare them with the original videos from the encoding step in order to investigate the influence of the different applied network configurations.

IV. Conclusion

With *StreamSim*, we introduced a flexible framework which separates the streaming functionality into five dedicated steps. The encoding and streaming steps are individually configurable. The framework is so far dedicated to the video streaming of H.264- and H.265-encoded video sequences over MPEG-TS/RTP/UDP, MPEG-TS/UDP or RTP/UDP. It allows simulating encoding impairments and transmission degradations such as packet loss, delay and jitter for various models of impairment distributions. Loss traces can be stored and reused. With this framework, video sequences with compression and transmission impairments typical for services such as IPTV can be generated so that the perceptual quality impact of such impairments can be studied and modeled.

The current framework implementation runs on Linux and is programmed in Python. It is made publicly and freely available with a detailed user documentation and developers guide. It can be found on https://gitlab.com/vqeg/streamsim.

References

[1] P. Hanhart, M. Rerabek, F. De Simone, and T. Ebrahimi, "Subjective quality evaluation of the upcoming HEVC video compression standard," in *Proc. of SPIE*, vol. 8499, 2012.

[2] B. Oztas, M. T. Pourazad, P. Nasiopoulos, and V. C. M. Leung, "A study on the HEVC performance over lossy networks," in *Proc. of 19th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2012, pp. 785–788.

[3] Y.-L. Chang, Y. Resnik, Z. Chen, and P. C. Cosman, "Motion compensated error concealment for HEVC based on block-merging and residual energy," in *Proc. of Packet Video Workshop (PV)*, 2013.

[4] J. Nightingale, Q. Wang, C. Grecos, and S. Goma, "Subjective evaluation of the effects of packet loss on HEVC encoded video streams," in *Proc. of IEEE Third International Conference on Consumer Electronics (ICCE)*, 2013, pp. 358–359.

[5] J. Nightingale, Q. Wang, and C. Grecos, "HEVStream: a framework for streaming and evaluation of high efficiency video coding (HEVC) content in loss-prone networks," in *Proc. of IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, 2012, pp. 404–412.

[6] T. Schierl, S. Wenger, Y.-K. Wang, and M. M. Hannuksela, "RTP Payload Format for H.265/HEVC Video," in *IETF Internet Draft*, 2012.