KSII The 11th International Conference on Internet (ICONI) 2019.

AVC/HEVC on Hardware and Software for Advancing Video Streaming System

Tuan Thanh Le[+] and Eun-Seok Ryu[++]
[+] Department of Computer Engineering, Gachon University
[++] Department of Computer Education, Sungkyunkwan University
South Korea
[e-mail: tuanlt@gc.gachon.ac.kr, esryu@skku.edu]
*Corresponding author: Eun-Seok Ryu

Abstract

Currently, High-Efficiency Video Coding (HEVC) has become the most promising video technology. However, the deployment of HEVC into video streaming system was limited by elements such as cost, the complexity of the design and the compatibility with current systems. While HEVC was considering deploying into various systems with many restrictions, H264-AVC can be the best option for current video streaming systems. This paper presents an adaptive method to manipulate video stream by using video coding on a designed integrated circuit (IC) with a private network processor. The proposed system provides a possibility to stream multimedia data from a camera or other video sources. For this work, a sequence of video/audio packets from video source are forwarded to designed IC, which called as transmitter Tx. The transmitter Tx processes input data into a real-time stream by using private protocol according to Real-Time Transport Protocol (RTP) for video/audio, then Tx transmits them to video client via the internet. Video client can include hardware decoder or software decoder to process received video stream. Tx uses video coding H264-AVC or HEC to encode multimedia data. By controlling the message exchanging between Tx and video client, the streaming can be established rapidly, and the streaming throughput can be achieved around 50 Mbps with latency approximate 80 msec.

Keywords: Video streaming, RTP, H.264, HEVC, video coding.

1. Introduction

In recent time, the HEVC [1] has gradually become the leading trend in the field of video processing. HEVC (H.265 / MPEG-H part2) is an emerging technology in the video coding field. Therefore, in order to support HEVC, video service platforms are upgrading to support HEVC coding. The first version of HEVC achieved an approximately 50% bitrate reduction compared to its predecessor H.264-AVC with equivalent subjective quality [2]. Over the past few years, many video transmission systems, CCTV systems, or online video services have gradually moved to the HEVC application, which increases bandwidth responsiveness. However, HEVC also brings many limitations such as high cost, system complexity, compatibility with predecessor video codecs. Therefore, H.264-AVC [3] codec application systems are still the mainstream today.

Video codec H.264-AVC allows the video processing system to encode/decode video streams at low cost in terms of system resources and processing time. The main drawback to H.264 AVC is that its bandwidth when compared to HEVC. Especially with live streaming systems, the limited bandwidth becomes even more urgent in the current multi-channel

This research was supported by the MSIT(Ministry of Science and ICT)

contexts. Therefore, to improve the performance of systems using H264-AVC codec, we propose a method to increase the transmission bandwidth based on the designed IC that comes with the H264-AVC codec inside. We used the designed IC with private network processor to efficiently embed the H264-AVC encoder/decoder. By developing the H264 encoder/decoder under kernel space in both server and client, we are able to minimize processing time thereby increasing overall system performance.

To ensure the performance of video streaming over the internet environment, we also implemented a protocol based on the Real-Time Transport Protocol (RTP) according to [4] and [5]. This customized protocol allows the transmission system to be able to meet real-time transmission with an average bitrate of approximately 50 Mbps. In addition, the development of encoder/decoder in Linux kernel space allows us to apply HEVC codec into designed IC similar to H264.

The remaining part of the paper is organized as follows: Section 2 presents the proposed method based on video coding point of view and RTP protocol. Section 3 shows the performance of the proposed system. Finally, section 4 concludes the efficiency of the proposed method and give further work.

2. Proposed Method

As shown in Fig. 1, the proposed method provides an efficient streaming system based on HDMI connection and a video encoder/decoder on designed IC (Tx, Rx). Video source such as PC, camera releases video frames into Tx device via HDMI communication. Then, Tx encoders (included H264 and H265 encoders) will encode these video picture into video bitstream. Therefore, each Network Abstract Layer (NAL) packet can be attached into the payload of modified RTP packet. To transmit modified RTP packet, we implemented a real-time transport protocol named as QRTP based on RTP [5]. Finally, Tx generates QRTP packet and transfers it to client-side.

In client-side, we provide support for two kinds of client: Rx (designed IC) and VLC player [6]. The design of Rx is almost same with Tx except for its decoder. Both Tx and Rx process video streaming by using private network processor, which was designed on our IC. Fig. 2 shows the functional flow of the proposed system in case using Tx and Rx. All procedures are deployed to kernel space to reduce processing time. By this way, Tx and Rx can provide a high-speed connection for video streaming. In case we use VLC player as a video streaming client, VLC player can pair to Tx as the same as Rx.

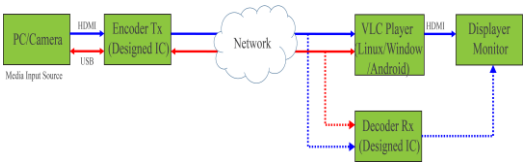


Fig. 1. The conceptual architecture of the proposed system

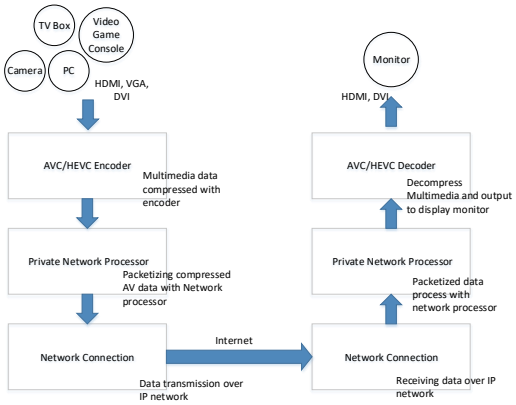


Fig. 2. Functional flow of system with Tx/Rx

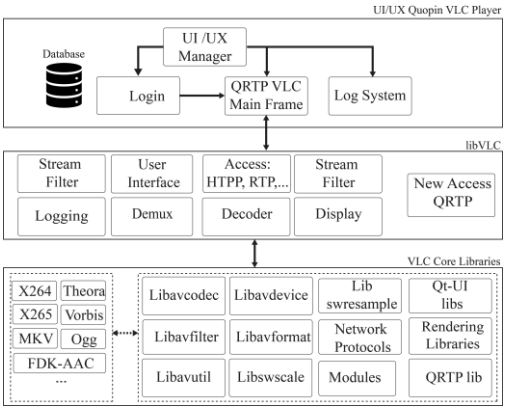


Fig. 3. VLC structure including QRTP protocol

As shown in Fig. 3, the VLC player was implemented within QRTP protocol support as access module [7]. Additionally, VLC library core also provides configurable options for QRTP similar to RTP. Fig. 4 shows the QRTP

protocol over UDP/IP stack. H.264/H.265 NAL packets are encapsulated as data part of QRTP packet. The total length of QRTP packet is around 22~1038 bytes. UDP header length is 8 bytes, IP header length is 20 bytes, ethernet header length is 14 bytes.

QRTP protocol was modified based on original RTP. Exactly, QRTP payload type (PT) map was designed differently to support both: pairing flow and video/audio streaming procedure as in table 1. PT value in table 1 is one byte in hexadecimal format. Pairing flow is the first procedure to exchange control message between Tx and client-side.

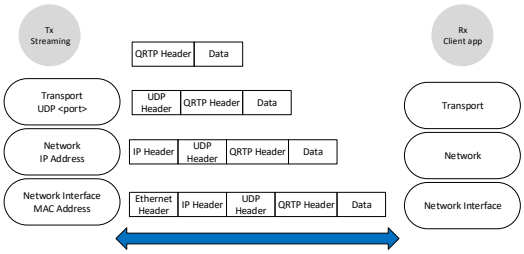


Fig. 4. Proposed protocol's stack

Table 1. Description of payload type of QRTP

Payload type	Description	Direction
0x71	Connection request	Rx->Tx
0x72	Connection response	Tx->Rx
0x78	Video resolution: vertical size	Tx->Rx
0x79	Video resolution: horizontal size	Tx->Rx
0x7A	Resolution ACK	Rx->Tx
0x7F	Video stream req.	Rx->Tx
0x00	Video stream data	Tx->Rx
0x03	Audio stream data	Tx->Rx
0x7C	Audio sampling rate	Tx->Rx
0x7B	Reset flow	Tx->Rx

The pairing flow procedure allows Tx and Rx establishing QRTP session before streaming video/audio. As shown in Fig. 5, the pairing flow can be concluded as follows:

1) Ping request/Ping reply: To check the internet route between hosts. Additionally, this step also helps Tx and client can verify correctly their partner.
2) Connection request/connection reply: Establishing a connection and ACK message exchanging.
3) Resolution information: Tx sends resolution detail double times for setting vertical and horizontal sizes.
4) Resolution ACK: Client sends message to confirm resolution setting to Tx.
5) Video stream request: Client sends a video stream request to Tx.
6) Video & audio data: Tx streams video and audio data to client.
7) Audio sampling rate: while Tx was streaming video/audio data, it sends audio sampling rate information to client. The client will conFig. its audio setting following message in details.

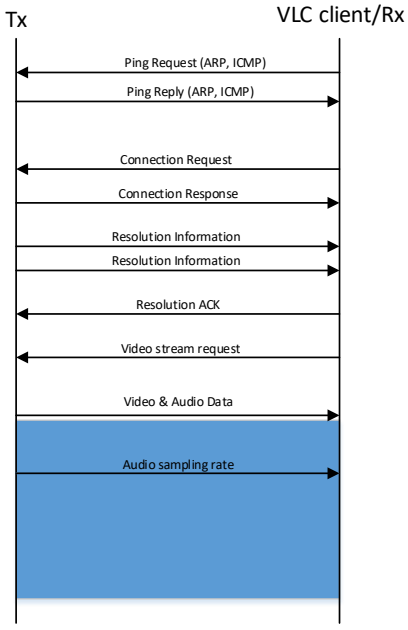


Fig. 5. Pairing flow between Tx and VLC client/Rx

3. Performance Evaluation

The proposed system may significantly demonstrate real performance when transmitting video content in real-time. In order to analyze the proposed method, we set up a testbed as illustrated in Fig. 6. We set up a PC with a Core

i5-7200U 2.5GHz processor, 16 GB of memory with Linux Ubuntu OS 18.04 GCC 7.4.0 as VLC client. We also set up a Raspberry Pi 2 Model B [8] as the video source. Additionally, we installed a Canon EOS 5D camera [9] as input video camcorder. Additionally, we use two designed ICs (Tx, Rx) as transmitter and receiver. The block diagram of working flow of Tx/Rx was illustrated as in Fig. 7.

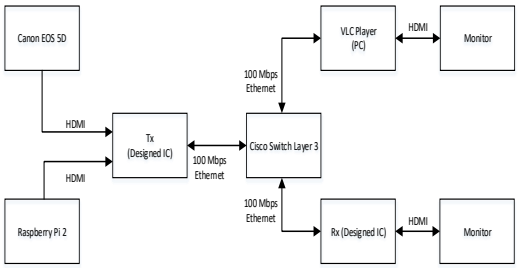
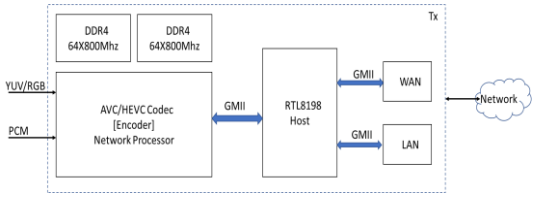
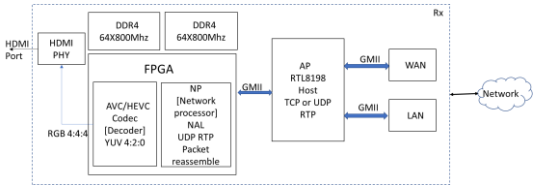


Fig. 6. The testbed's scenario

Regarding software, we used FFmpeg software with add-on libraries [10] as an evaluation tool. Besides that, the VLC software version 3.0.8 was used as video player of PC client. Video encoder/decoder H.264 support resolution from 16x16 (minimum) to 1920x1080 (maximum). Video codec H.264 provides supporting for base, main and high profile within level 4.2.



(a)



(b)

Fig. 7. (a) Tx encoder block diagram; (b) Rx decoder block diagram.

As shown in Fig. 8, the bitrate of the video stream is around 49 Mbps with 1920x1080 video resolution, and 18 Mbps with 1080x720 video resolution. Table 2 shows that the latency for maximum resolution 1080p is always less than 82 msec. In case lower video resolution, the latency is smaller than 82 msec. The average latency is approximately around 80 msec.

To verify the quality of the decoder, the comparison between reconstructed video at the client and the original video was progressed. As shown in Table 3, Y channel PSNR values show the performing unsatisfactory compared to other quality metrics when it comes to estimating the quality of images and videos as perceived by humans. We verified the comparison for both 1080p and 720p videos. All Y-PSNR values are higher than 38 (dB) that the quality of reconstructed videos at the client is reasonable to feel fully immersed in video streaming and videos as perceived by humans.

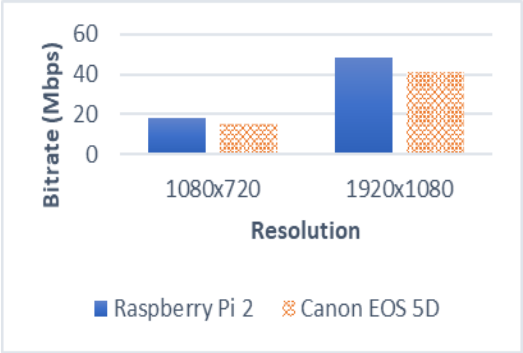


Fig. 8. The bitrate of video streaming: proposed system using H.264-AVC

Table 2. Latency evaluation

	Resolution 1080x720 (720p)	Resolution 1920x1080 (1080p)
Raspberry Pi 2	71.36 msec	78.93 msec
Canon EOS 5D	73.47 msec	81.52 msec

Table 3. Y-PSNR comparison

	Resolution 720p Y-PSNR	Resolution 1080p Y-PSNR
Raspberry Pi 2	41.46	40.19
Canon EOS 5D	38.25	38.03

4. Conclusions

This paper presents an adaptive video streaming system by using efficiently video codec on both hardware and software. The experimental results show that the proposed system can provide high performance for video streaming services in real-time with a throughput of approximately 50 Mbps and latency around 80 msec. Furthermore, the proposed method also allows improving the video streaming system in some important tasks such as upgrading HEVC video codec, extending service by adding control messages and more. In the future, building a standard library and releasing packaged software are two main keys to our video streaming system.

Acknowledgment

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2019-2017-0-01630) supervised by the IITP(Institute for Information & communications Technology Promotion).

References

[1] JCT-VC, "High-Efficiency Video Coding". Available: https://hevc.hhi.fraunhofer.de/

[2] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video 344 coding standards—Including High-Efficiency Video Coding (HEVC)", *IEEE Trans. Circuits Syst. Video Technol*., 345 vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[3] ST. Wiegand and Gary J. Sullivan, "Overview of the H.264/AVC Video Coding Standard", *IEEE Transactions on 349 Circuits and Systems for Video Technology*, Vol.13, No.7, p.p 560 - 576, IEEE, July.2003.

[4] IETF, "Real-Time Transport Protocol RFC 1889". Available: https://tools.ietf.org/html/rfc1889

[5] IETF, "RTP Profile for Audio and Video Conferences with Minimal Control RFC 1890". Available: https://tools.ietf.org/html/rfc1890

[6] VLC, "Video Lan Player". Available: https://www.videolan.org/vlc/index.html

[7] VLC, "VLC core library – modules". Available: https://wiki.videolan.org/Documentation:Modules/

[8] "Raspberry Pi 2 model B product". Available [Online]: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[9] Canon, "Canon EOS 5D Mark IV". [Online]. Available: https://www.usa.canon.com/internet/portal/us/home/products/details/cameras/eos-dslr-and-mirrorless-cameras/dslr/eos-5d-mark-iv

[10] FFMPEG, "FFmpeg software version 4.0.2 and document guide". [online]. Available: https://ffmpeg.org/download.html#get-sources