Low Latency Live Streaming System with Congestion Control

Toya Kinoshita and Hiroyuki Hisamatsu

***Abstract*—In recent years, the traffic for live streaming on the web has been increasing. The current live streaming methods that use MPEG-DASH or HLS are simple and scale easily to many clients using HTTP. However, they do not take into account the communication between the distributor and the viewer. As a result, latency between the distributor and the viewer is relatively high. Therefore, in this paper, we propose a low latency live streaming system on the web using WebRTC. Since WebRTC uses UDP, it does not have a congestion control mechanism. Depending on the network congestion, it is possible to stream video with quality that exceeds the available bandwidth. Therefore, we propose a system to change the video quality based on the congestion status. The proposed system increases or decreases the video transfer rate by changing the quality of the streamed video depending on the network conditions. We have evaluated the proposed system in a real network environment. As a result, we showed that the delay of the proposed system is smaller than that of the MPEG-DASH system. We also showed that the proposed system can change the quality of the video and switch the transmission rate appropriately according to the network conditions.**

***Index Terms*—Low latency video streaming, MPEG-DASH, WebRTC.**

I. Introduction

It is becoming more and more common to view video media using a web browser. In video sharing services such as YouTube and Video on Demand (VoD), the progressive download method is commonly used to play video files while downloading them over HTTP. MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [1] and HTTP Live Streaming (HLS) [2] are video streaming protocols over HTTP protocol. These protocols do not require special servers such as Adobe Media Server but can realize adaptive streaming where the quality of the video is changed according to the network conditions using only an HTTP server. In addition, they can use Content Delivery Network (CDN) to deliver content over a streamlined network.

In addition to video streaming, "live" streaming of video in real time is also becoming more common [3]. Live streaming by users on the Internet is growing in scale and demand. Web services such as YouTube Live and Twitch.tv, as well as smartphone applications, are increasing the opportunities for individuals to live stream. Unlike TV programs or videos such as VoD, live streaming often involves communicating with an unspecified number of viewers in real time via chat or voice. MPEG-DASH and HLS divide the video stream into segments of several seconds, and the viewer reads these files according to the manifest file. The length of each media segment is typically between 2 and 10 seconds. There is a delay for transcoding and for the segment to be placed on the server or CDN. As a result, there is a delay of about 10 seconds or more until it reaches the viewer. MPEG-DASH and HLS are simple as protocols and easy to scale, but they are likely to have high latency when used for live streaming.

In general, a delay of 10 seconds or more makes communication between the broadcaster and the viewer difficult and prevents natural communication in live streaming [4]. Using Flash Player, low-latency live streaming is possible using Real-Time Messaging Protocol (RTMP) [5]. However, the developer of Flash Player announced in 2017 that it would no longer be supported, and it actually ended in 2020 [6]. Since it has become an obsolete technology, it is practically impossible to use Flash Player in browsers and smartphones.

Therefore, in this paper, we propose a system for realizing low latency live streaming with WebRTC, which is the current browser standard by utilizing the publishing method of the media encoder currently in use. Specifically, the proposed system uses the Real-time Transport Protocol (RTP) [7], which is a protocol used to exchange video in the media channel of WebRTC, to deliver video in real time between the server and the viewer. By using RTP, the proposed system can achieve a low latency media delivery because the video is directly streamed in our system, unlike segmented video delivery such as MPEG-DASH.

In addition, RTP uses UDP as its transport protocol. UDP has no congestion control mechanism and is suitable for real-time communication. When the network is congested, communication using UDP can cause worsening congestion, which can be a major problem. Furthermore, if a high quality video that requires a lot of bandwidth is transferred when the network is congested, a large number of information will be lost during the transfer, and the viewer will not be able to play it back, making such a transfer meaningless. In the proposed system, congestion control is performed at the application layer to solve this problem. Specifically, the system changes the video quality according to the network congestion status to achieve appropriate video quality transmission. In addition, we will evaluate the proposed system in a real network environment to show its effectiveness.

This paper is organized as follows. First, we describe related works in Section II. Next, in Section III, we show the problem of delay in MPEG-DASH. Then, in Section IV, we describe the proposed system. Then, we evaluate the proposed system in a real network in Section V. Finally, in

Manuscript received January 5, 2021; revised February 25, 2021.

Toya Kinoshita and Hiroyuki Hisamatsu are with the Graduate School of Information Science and Arts Osaka Electro-Communication University, 1130-70 Kiyotaki, Shijonawate, Osaka, Japan (e-mail: mt20a003@oecu.jp, hisamatu@osakac.ac.jp).

doi: 10.18178/jacn.2021.9.1.280

Section VI, we summarize this paper and discuss future work.

II. Related Research

There are already several studies on WebRTC and MPEG-DASH [8]-[12]. First, in [8], a low latency streaming method using WebRTC for file transfer is proposed and evaluated by simulation, while using the MPEG-DASH framework. In this paper, the authors focus on the problems of TCP in file transfer, and do not consider the problems of segmentation in MPEG-DASH. In this paper, we eliminate the delay caused by segmentation by directly streaming the video using a media channel.

In [9], a UDP-based transmission system using the WebRTC media channel is implemented and its Quality Of Experience (QoE) is evaluated. The evaluation results show that the QoE is better than that using MPEG-DASH in most cases, but the QoE decreases when the network load is high. The delay between the broadcaster and viewer, which is the focus of this paper, is not considered, nor is congestion control.

In [10], a framework for testing applications using WebRTC is proposed. As an example, they show that it is possible to evaluate the bandwidth and latency when the number of clients increases when WebRTC is used for delivery. It does not consider that WebRTC has no congestion control.

In [11], streaming using MPEG-DASH, RTSP, and RTMP in mobile and wireless networks is evaluated in the view of QoE. The evaluation results show that RTSP has the shortest time to start playback, but the QoE decreases when packet rejection occurs. However, RTSP and RTMP are currently not practical to use in browsers. In addition, MPEG-DASH suffers from relatively large latency problems, which causes problems for two-way communication in live streaming.

In [12], the authors evaluate the case of live streaming over HTTP. The results of the evaluation show that in order to reduce the delay in reaching the viewer from the publisher of the live streaming, it is necessary to reduce the segment size, and at the same time, if the playback time of the segment is reduced to about 1 second, the resulting delay increases by about 2 seconds. In this paper, unlike [12], we use WebRTC to stream the video directly without segmentation.

In summary, previous studies have considered either MPEG-DASH-based methods or RTMP/RTSP-based methods that are not available in browsers. There are also studies using WebRTC, but they do not evaluate the delay and do not take congestion control into account. The method we propose in this paper is very different in that it can be used with a web browser, and it takes into account the video delay and congestion control between the broadcaster and the viewer.

III. Delay of Streaming Using MPEG-DASH

MPEG-DASH is a streaming method compatible with the current Internet browser standard HTML5 [13] and is used in video services such as YouTube and Netflix [14]. MPEG-DASH is currently the most used streaming method. In this section, we show the delay and its cause in the adaptive streaming method of MPEG-DASH over HTTP.

In general, low latency communication is needed when communicating with someone in a remote location. An existing protocol for real-time media transfer is RTMP, which uses TCP to deliver media with low latency. Currently, it is generally possible to deliver video from a broadcaster to a server using RTMP. However, RTMP cannot be used for transmission from the server to the viewer at present, and MPEG-DASH is used in most cases. RTMP is a protocol that is not supported by browsers themselves. Until now, viewers have been able to use communication using RTMP from their browsers by using Adobe Flash Player. However, since the development of Flash Player has been terminated, continuing to use Flash Player poses a security problem. Therefore, it can be said that RTMP cannot be used in browsers at present.

Therefore, MPEG-DASH, which can be used based on HTTP, has been used for streaming between servers and viewers. Fig. 1 shows the workflow of streaming using MPEG-DASH. The broadcaster uses video streaming software to send the video to the server in RTMP, and the server's encoder generates MPEG-DASH video segments. In MPEG-DASH, the video stream is divided and stored into video files called segments, and information such as bit rate, resolution, segment time, and path are stored in an XML format file called Media Presentation Description (MPD). The MPD file is placed on an ordinary HTTP server, and the viewer retrieves the MPD file. The viewer can then retrieve the media according to the information in the MPD file.

Fig. 2 shows the delay for segment transmission using MPEG-DASH. In the network, there are propagation delays between the broadcaster and the server, and between the server and the viewer, T_{n_1} and T_{n_2} , respectively. The beginning of the media arrives at the server T_{n_1} after the broadcaster starts distribution. The server waits for the video from the broadcaster to arrive for the specified time T_s of the segment, and then writes out the segment. The time taken to write out the segment is T_w. In addition, the transmission of a segment takes T_μ. T_μ is determined by the segment size and the bandwidth between the server and the viewer. To ensure uninterrupted playback, the video data is buffered at the viewer side, and there is a delay of T_b from the time the segment arrives at the viewer until it is played. The time T of the delay until the start of playback is shown in the following equation.

$$T = T_{n1} + T_s + T_w + T_{n2} + T_\mu + T_b$$

Propagation delay T_{n_1} and T_{n_2} can be shortened by using CDN. T_w is very small, and T_μ is not a problem when the network bandwidth is large. T_b is a trade-off between uninterrupted playback and delay. If small delay is important, it can be reduced.

However, it is difficult to shorten the waiting time T_s for the arrival of the video data for one segment. It may be thought that the delay can be reduced by reducing the segment length, but this is not true. When the segment length is reduced, the performance of transfer efficiency decreases due to the increase in IO for file writing and the increase in communication overhead such as headers due to the increase in the number of requests in the HTTP network. As a result, it may cause more delay [15], [16]. In current live streaming

systems that uses MPEG-DASH, it is difficult to reduce the delay T_s , and there is a limit to the reduction of the overall system delay.

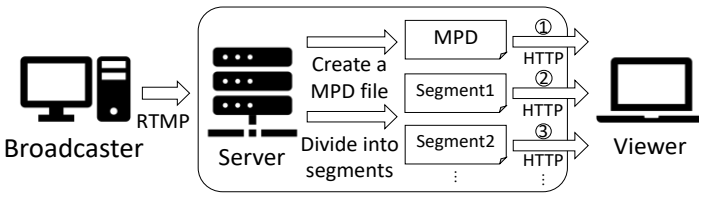

Fig. 1. Workflow of streaming using MPEG-DASH.

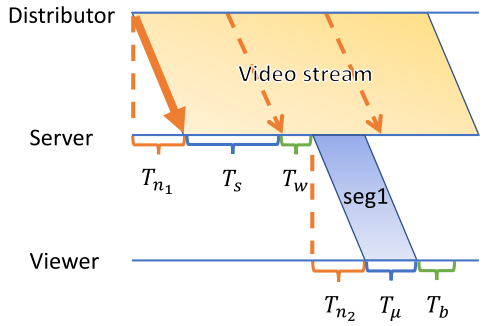

Fig. 2. Delay for segment transmission using MPEG-DASH.

IV. Design of System

In this section, we describe the streaming mechanism and the congestion control mechanism of our proposed system.

A. Streaming Mechanism

Fig. 3 shows an overview of the proposed system. The broadcaster sends the video generated by the broadcaster to the server using RTMP, and the server delivers the video to the viewer via WebRTC. The broadcaster encodes the video using an encoder and publishes the stream to the server using RTMP for live streaming. RMTP is the standard posting protocol for live streaming and is supported by major software such as FFmpeg [17] and Open Broadcaster Software (OBS) [18] and is used by services such as YouTube and Twitch.tv for posting video stream from broadcasters.

WebRTC is a method of real-time communication using a web browser. In our proposed system, WebRTC is used to send video from the server to the viewer with low latency. Fig. 4 shows the flow of starting a connection between the server and the viewer in WebRTC. The Session Description Protocol (SDP) data required for the WebRTC connection between the server and the browser, as well as the channel data for the stream to be viewed, is exchanged between the two parties using WebSocket. The two-way connection can be used to communicate video information in addition to the connection, and we use this for congestion control.

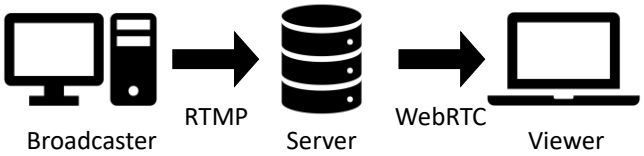

Fig. 3. Outline of proposed system.

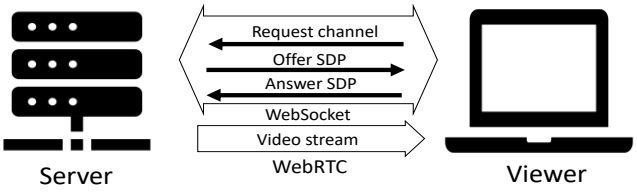

Fig. 4. WebRTC connection.

B. Congestion Control Mechanism

When the network connection of the viewer of the live streaming is slow or the network condition is unstable, the original stream becomes unstable and cannot be viewed because the video stops playing. In the proposed method, the server transcodes the live video received from the broadcaster to multiple lower quality channels, which are different from the original. Depending on the reception status of the stream at the viewer's side, the channel is changed to the appropriate one. The delay caused by transcoding is smaller than that of a method that writes to segments because the video is streamed without being written to a file.

Between the server and the viewer, streaming is performed using WebRTC, but its transport layer protocol is UDP, which does not have a congestion control mechanism. Therefore, there is no congestion control between the server and the viewer, which can make congestion even worse when network is congested. From the perspective of all network users, congestion control is necessary for streaming. In addition, when the network is congested, even if high quality video is transmitted in excess of the available bandwidth, the many parts of video stream will be rejected by the network and the viewer will not be able to play them.

When the proposed system determines that the network is congested, it reduces the transmission rate by switching to a lower bit rate video. When the system judges that there is enough available bandwidth in the network, it switches to a higher bit rate video. Specifically, the frame rate of the received video and the estimated TCP throughput are used to determine the network congestion status and control the transfer rate from the server to the viewer. If the frame rate of the video being played by the viewer is lower than a certain value, the proposed system considers that congestion is occurring and reduces the transmission rate. If the frame rate is higher than a certain value and the bit rate of the higher quality video is higher than the current estimated value of TCP throughput, the network bandwidth is considered to be sufficient and the transfer rate is increased.

From the perspective of QoE, uninterrupted playback is considered to be the most important aspect of live streaming. For this reason, the proposed system quickly responds to network congestion that may cause interruptions in playback. On the other hand, if the network congestion is resolved, the response, switching to a higher quality stream, will take a relatively long time.

The operation of the proposed system is described as a discrete-time system with unit time Δ. The proposed system judges the network congestion status every Δ. First, if the frame rate f_c of the video currently being played by the viewer is less than the default rate F_t the system considers that the network is congested and requests a lower level video stream from the server. If f_c is greater equal than F_t and if

$f_c \geq F_t$ for more than N_{up} consecutive times in this judgment, the estimated TCP transfer rate λ is compared with the bit rate of the higher level video. If λ is not smaller, the network bandwidth is considered to be sufficient to stream the higher-level video, and a request for the higher-level video stream is made to the server. The estimated value of the TCP transfer rate λ is given by the following equation [19].

$$\lambda = \frac{s}{R\left(\sqrt{\frac{2p}{3}} + 12\sqrt{\frac{3p}{8}}p(1+32p^2)\right)}$$

where R is the round-trip time, p is the packet loss probability, and s is the packet size.

Fig. 5 shows the workflow for changing the channel. When a change in the network state is detected, the viewer requests another channel from the server over WebSocket. When the server receives the request for a lower channel, it switches to the lower channel, if it exists. If it has already sent the lowest channel, it does not do anything. When a higher level channel request is received, as with the lower level channel request, if a higher level channel exists, it is switched to, otherwise nothing is done. We summarize the definitions of the symbols used in Table I.

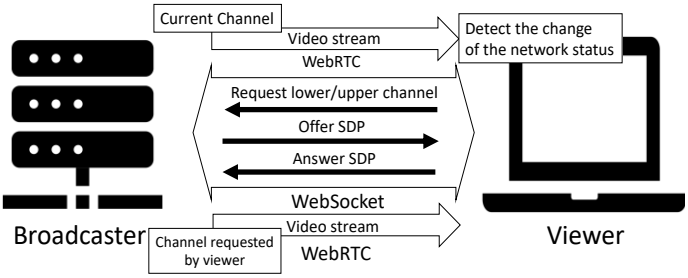

Fig. 5. Workflow for switching the channel.

TABLE I: DEFINITION OF SYMBOLS

f_c	Frame rate of the video that the viewer is playing.
F_t	Frame rate threshold
N_{up}	Threshold for number of consecutive for $f_c \geq F_t$
λ	Estimated TCP transfer rate.

V. EVALUATION OF THE PROPOSED SYSTEM

In this section, we implement the proposed system and evaluate it in a real network. Specifically, we evaluate the delay between the broadcaster and the viewer of the proposed system, and its behavior when the network condition changes.

A. Stream Delays

Fig. 6 shows the network environment used for the delay evaluation. The broadcaster, server, and viewer exist in the network, and the broadcaster and viewer are in the same host. The server is located on the Internet, making it a real Internet delivery environment. The average round-trip time between the local host and server was 23.744 ms, and the bandwidth from the server to the localhost server was 33.9 Mbits/s.

We used h264 as the codec for the video stream, which is the most supported by browsers, and x264 as the encoder. For the encoder options, we used the zerolatency tune for low latency delivery and set the keyframe to 1. For the encoder used for transcoding in the server, we used the superfast profile to reduce the overhead of transcoding. This can be changed when real-time encoding is possible, even if image quality is important, depending on the calculation speed of the server performing transcoding.

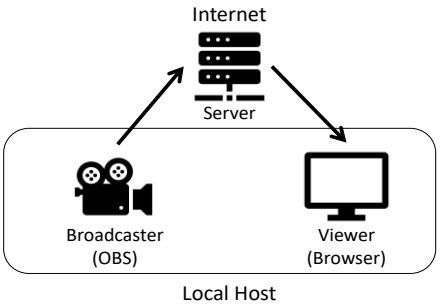

Fig. 6. Network environment used for the delay evaluation.

In this evaluation, the original stream and the lower streams, 720p and 480p, will be transcoded and provided. The other streamable channels are 1080p, 360p and 160p. We summarize the stream channels of the proposed system in Table II.

TABLE II: OFFERED CHANNEL

Channel Name	Resolution	Bit rate
live	-	-
1080p	1920 x 1080	3000 kbps
720p	1280 x 720	2000 kbps
480p	888 x 480	1200 kbps
360p	640 x 360	560 kbps
160p	284 x 160	180 kbps

The broadcaster publishes the stream to the server using OBS with the bit rate set to 2500 kbps CBR and the frame rate set to 30 fps. For transcoding, GStreamer [20] is used. The segment length for MPEG-DASH is set to the default value of 10 seconds.

First, we measure the delay in the delivery of the video from the broadcaster to the viewer. This is done by calculating the difference between the frame number output by OBS and the frame number displayed on the client. Fig. 7 shows an example of the displayed frame number. The screen on the left side of the figure shows the video being transmitted by the broadcaster, and the one on the right side shows the video being viewed by the client. The numbers displayed on each screen are counters that are incremented for each frame; a difference of 30 frames indicates a delay of 1 second. In the case of Fig. 7, there is a difference of 39 frames between the broadcaster and the viewer, indicating the delay of 1300ms.

We measure the latency for one minute from 5 minutes after the start of transferring. We measure the delay in two channel, “live” and “480p”. We also conduct the evaluation of the delay when MPEG-DASH is used in the part of transferring from the server to the viewer. Fig. 8 shows the distribution of the delays for the proposed system and video streaming by using MPEG-DASH.

From this figure, we can see that the delay of MPEG-DASH is large and that the delay of the proposed method is

very small. This is because MPEG-DASH has a large delay due to the segmentation in the server. On the other hand, the proposed method can transfer the video as a stream between the server and the broadcaster without segmentation by using WebRTC. As a result, this low latency can be achieved. Also, the delay between the lower channel and the original channel is about one second, which is caused by transcoding.

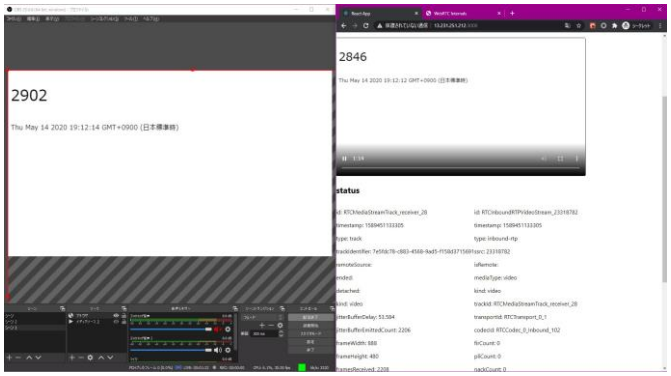

Fig. 7. Screens of the broadcaster and the viewer.

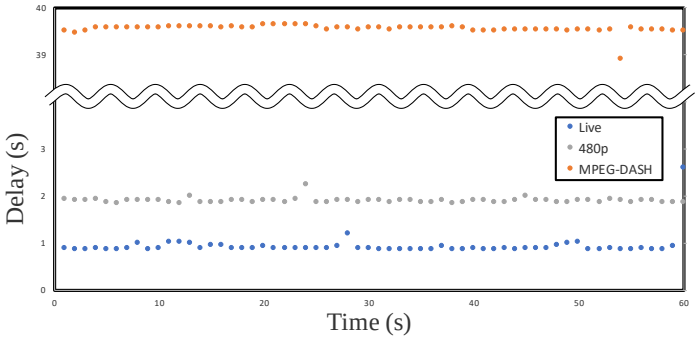

Fig. 8. Delay between the broadcaster and the viewer.

B. Response to Changes in the State of the Network

In a real network, we evaluate the behavior of the proposed system to changes in network congestion. Fig. 9 shows the network environment used for the evaluation. The network contains a broadcaster, a server, and viewers, as well as an HTTP server and clients to load the bandwidth between Router 1 and Router 2. 180 second experiments are conducted, and live streaming is performed from the beginning to the end of the experiment. From 60 seconds to 120 seconds after the start of the experiment, we load the bandwidth between Router 1 and Router 2 with HTTP traffic. Specifically, after 60 seconds, 10 HTTP file transfers are started, and after 120 seconds, all HTTP file transfers are terminated. In the evaluation, we set the parameters of the proposed systemΔ to 1s, the frame rate of the video streamed by the server to the viewer to 30fps, F_t to 24fps, N_{up} to 5 and s to 1460Kbyte.

Fig. 10 shows the frame rate of the video played by the viewer and the receive rate of the video when the available bandwidth of the network changes during the experiment. The frame rate and receive rate are stable until 60 seconds after the start of the experiment. From 60 seconds after the start of the experiment, when the network bandwidth is overloaded by HTTP traffic, the frame cannot be decoded and the frame rate drops. Immediately, the proposed system judged that the network was unstable and changed the channel from live (bit rate: 6Mbit/s) to 720p (bit rate: 2Mbit/s). As a result, the bit rate of the video is reduced and the reception speed of the video is reduced. As a result of lowering the video bit rate, the data necessary for playback can be received, and the frame rate has returned to 30 fps.

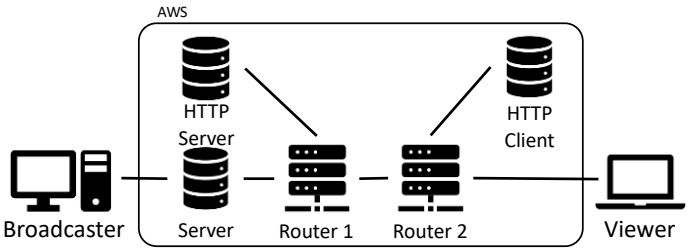

Fig. 9. Network environment used for the behavior evaluation of the response to changes in network congestion.

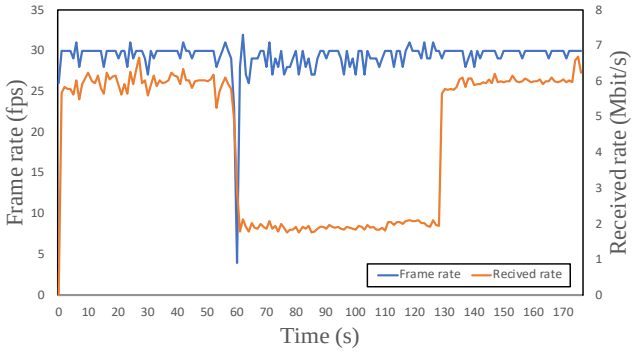

Fig. 10. Frame rate of the video being played back and receive rate at the viewer.

In addition, 120 seconds after the start of the experiment, the network load disappears, and another 5 seconds ($5=\Delta\times N_{up}=1\times5$) after, the system judges that there is room in the network bandwidth, and the channel is changed from 720p to live, resulting in a higher video receive rate. As a result, we can confirm that the proposed system is working properly according to the network condition.

VI. Conclusion and Future Work

In this paper, we explain that the existing streaming method using MPEG-DASH has a large delay between the broadcaster and the viewer, making communication difficult, and propose a method to solve this problem. The proposed method achieves low latency by using WebRTC for the communication between the server and the viewer. In addition, the proposed method uses the frame rate of the video being played and the estimated TCP transfer rate to determine the network congestion and change the stream quality.

We implemented the proposed system and conducted evaluation experiments in a real network. As a result, we found that the delay between the broadcaster and the viewer of the proposed system is small, and that the delay due to transcoding is about one second. In addition, we showed that the proposed system can play back video of appropriate quality depending on the network status by judging the network congestion status using the frame rate of the playing video and the estimated TCP transfer rate.

In the future, we would like to evaluate the QoE of the proposed system and also evaluate it in a mobile network environment. In addition, we would like to compare the

proposed system with MPEG-DASH streaming when using CDN.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

HH designed research; TK performed research; TK wrote the initial paper; HH reviewed and edited the paper; all authors had approved the final version.

REFERENCES

[1] MPEG-DASH. [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-dash

[2] R. Pantos and W. May, "HTTP live streaming," *Request for Comments (RFC) 8216*, 2011.

[3] Iab. (2018). Live video streaming: A global perspective. [Online]. Available: https://www.iab.com/insights/ live-video-streaming-2018

[4] P. Cluff. (2019). The community gave us low-latency live streaming. then apple took it away. [Online]. Available: https://mux. com/blog/the-community-gave-us-low-latency-live-streaming-then-apple-took-it-away/

[5] H. S. Parmar and M. C. Thornburgh. (2012). Adobe's real time messaging protocol. [Online]. Available: https://wwwimages2. adobe.com/content/dam/acom/en/devnet/rtmp/pdf/rtmp_specification_1.0. pdf

[6] Adobe. (2017). Flash & the future of interactive content. [Online]. Available: https://theblog.adobe.com/ adobe-flash-update/

[7] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," *RFC 3550*, July 2003.

[8] S. Zhao, Z. Li, and D. Medhi, "Low delay MPEG DASH streaming over the WebRTC data channel," in *Proc. 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2016, pp. 1-6.

[9] Y. Maehara and T. Nunome, "Webrtc-based multi-view video and audio transmission and its qoe," in *Proc. 2019 International Conf. on Information Networking (ICOIN)*, 2019, pp. 181-186.

[10] B. Garcia, L. Lopez-Fernandez, F. Gortazar, and M. Gallego, "Analysis of video quality and end-to-end latency in WebRTC," in *Proc. 2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1-6.

[11] A. Aloman, A. I. Ispas, P. Ciotirnae, R. Sanchez-Iborra, and M. D. Cano, "Performance evaluation of video streaming using MPEG DASH, RTSP, and RTMP in mobile networks," in *Proc. 2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2015, pp. 144-151.

[12] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive http streaming of live content," in *Proc. 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2011, pp. 1-8.

[13] WHATWG. (May 2020). Html standard. [Online]. Available: https://html.spec.whatwg.org/multipage/

[14] C. Müller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer, "Using scalable video coding for dynamic adaptive streaming over http in mobile environments," in *Proc. 2012 the 20th European Signal Processing Conference (EUSIPCO)*, 2012, pp. 2208-2212.

[15] A. Sideris, E. Markakis, N. Zotos, E. Pallis, and C. Skianis, "MPEG-DASH users' QoE: The segment du- ration effect," in *Proc. 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, 2015, pp. 1-6.

[16] N. Bouzakaria, C. Concolato, and J. Le Feuvre, "Overhead and performance of low latency live streaming using MPEG-DASH," in *Proc. the 5th International Conference on Information, Intelligence, Systems and Applications (IISA 2014)*, 2014, pp. 92-97.

[17] FFmpeg. [Online]. Available: https://ffmpeg.org/

[18] OBS studio. [Online]. Available: https://obsproject.com/

[19] M. J. Handley, J. Padhye, S. Floyd, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specifica- tion," *Request for Comments (RFC) 5348*, 2008.

[20] GStreamer. Open source multimedia framework. [Online]. Available: https://gstreamer.freedesktop. org

Toya Kinoshita received the B.E degrees from Osaka Electro-Communication University, Japan, in 2019. He is currently a master's student at the Graduate School of Computer Science and Arts, Osaka Electro-Communication University. His research interests include network performance evaluation, TCP protocol design and evaluation. He is a student member of IEICE.

Hiroyuki Hisamatsu received M.E. and Ph.D. degrees from Osaka University, Japan, in 2003 and 2006, respectively. He is currently an associate professor of the Department of Computer Science, Osaka Electro-Communication University. His research work is in the area of performance evaluation of TCP/IP networks. He is a member of IEICE.