Lecture Notes in Electrical Engineering 376

Kuinam J. Kim
Nikolai Joukov
*Editors*

# Information Science and Applications (ICISA) 2016

Springer

# Contents

# Load Balancing Methodology for Efficient Real-Time Streaming Service Based on WebRTC

**Jisue Kim, Linh Ma Van and Jinsul Kim**

**Abstract** Recently, we propose a method to provide a Load Balancing WebRTC-based real-time streaming service to the user effectively. We will implement a real-time streaming service WebRTC for large number of smart devices, and Load Balancing method for distributed processing of large amounts of traffic through the OPNET simulation. In the experiments, we have analyzed the performance of the proposed method to prove the superiority of Load Balancing method.

## 1 Introduction

Recently, due to the increased amount of smart devices and smart TV devices, real time streaming services marketplace is growing too. In addition, existing streaming services are offers one-way interactive services. Two-way outstanding streaming services WebRTC currently is real-time communication standard technology that can send and receive voice and video in real time without any additional plug-ins using HTML5 and JavaScript API. Most of the existing release service requires a separate application or plugin but WebRTC is able to control the audio and video without any requirement, it's only need a Web browser. Additionally, users who use the streaming service WebRTC is a P2P network environment for transferring data with each other. However, with a large number of users in the same network environment in real time if you send or receive large amounts of data traffic, and the quality of service required by the user, and it is hard to satisfy the stability of the network. In this paper, we study a method of

J. Kim · L. Ma Van · J. Kim(✉)
School of Electronics & Computer Engineering, Chonnam National University,
77 Yongbong-ro, Buk-gu, Gwangju 500-757, Republic of Korea
e-mail: dyrk10@gmail.com, Linh.mavan@gmail.com, jsworld@chonnam.ac.kr



K.J. Kim and N. Joukov (eds.), *Information Science and Applications (ICISA) 2016*,
Lecture Notes in Electrical Engineering 376,
DOI: 10.1007/978-981-10-0557-2_125

controlling a Load Balancing to ensure the quality of service requested by the user in WebRTC environment.

## 2 Related Work

Because of the advantage of WebRTC, establishing a direct communication channel between two browsers without relaying the data through a web server. The possibility of establishing peer-to-peer channels between two browsers and the expected broad deployment opens the opportunity for new use cases that were not possible until now. Many researchers are involved in this field.

A distributed content sharing facility using WebRTC Data Channels as well as an emulation component[1] with an API to store content in and retrieve content from the underlying DHT. An overlay network for interactive multimedia communications[2]. They organized the overlay in four planes for data delivery, signaling, control and floor control. The overlay also includes self-organization techniques, re-organizing the clusters when clients leave or join. Furthermore, in case of Telemedicine[3], because of several emergency telemedicine now specialized, they designed a system just requires a mobile device with internet connection with either chrome or Mozilla browser installed in it. The device is carried within the ambulance to conduct an initial assessment of the patient and later brought to the nearest health center where further treatment is carried under the assistance of specialists whose telepresence is provided by WebRTC enabled devices. The implementation of a WebRTC gateway service that can forward ad-hoc RTP data plane traffic from a browser on one local network to a browser on another local network in case of Restricted Networks[4], was done by redirecting traffic from multiple concurrent streams securely between local gateway services that connect to it in The public gateway. In addition, Web Real-Time Communication (WebRTC) is used for solving problem of Web-based VC which has been the difficulty of implementing some advanced functions such as direct communications among users and file sharing.

## 3 Real-Time Streaming Using WebRTC and Load Balancing

In the Real-time Streaming Service, each user use WebRTC to send and receive video and audio in both directions. Load Balancing is a proposal method in WebRTC environment to give the data efficiently.

### *3.1 System Design*

Figure 1 shows an overall system architecture for real-time streaming services using WebRTC and Load Balancing. First, a user starts for real-time streaming

services then others can connect to the user via a Chrome or Firefox browser and connect to the Web Server. In the process of connecting to a Web Server, Signaling helps to establish the connection to the other user. Signaling processor is a coordinated communication to send and receive streaming data between users. After the process in Signaling is ended, users send and receive data in real time over the Load Balancer.

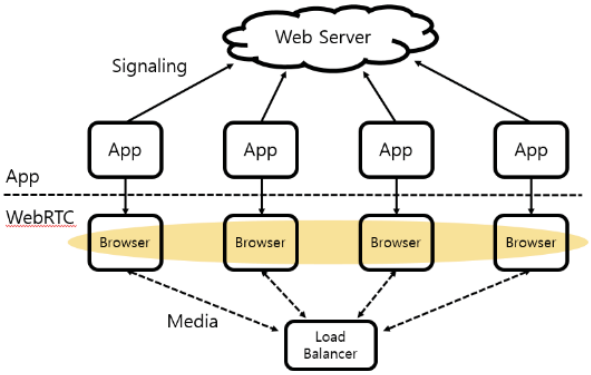



**Fig. 1** System Design using WebRTC and Load Balancing

## *3.2 WebRTC API*

WebRTC API provides three types of (MediaStream, RTCPeerConnection, RTCDataChannel). First, MediaStream streams audio and video tracks are received from a camera and microphone. RTCPeerConnection is a function for encryption, and bandwidth management, which is WebRTC component that processes the data reliable and efficient communication between the users. Finally RTCDataChannel operates with RTCPeerConnection API, to enable the P2P connection.

## *3.3 Implementation*

Figure 2 shows the real-time streaming services through WebRTC. We have four devices which are Tablet, Notebook, PC, and Phone. All of them is established the connection via the IP 168.131.39.48:3000. Each of these devices sent and received video and voice data in real time via P2P communications which are showed in Web browser (Chrome 28.6).

## *3.4 Load Balancing for Efficient Data Transmission*

In a case of large amount of traffic, a load balancing technique is applied to improve the quality of service, reducing the overload of the network system.

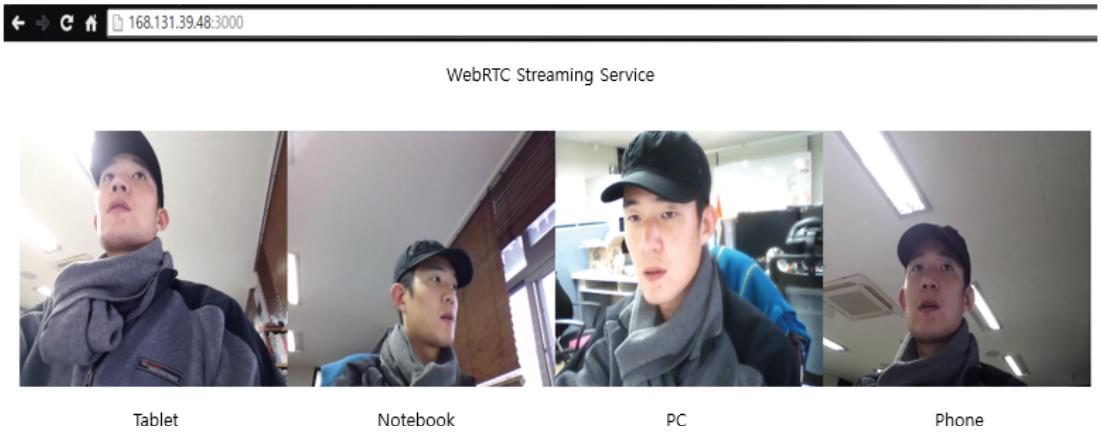



**Fig. 2** Real-time Streaming using WebRTC

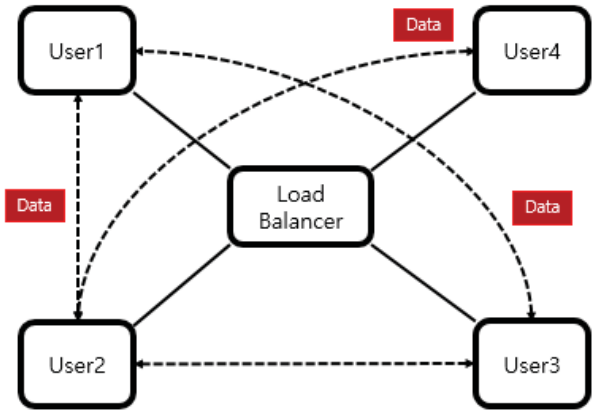



**Fig. 3** Load Balancing Flow

Figure 3 shows the Load Balancing Flow to reduce the overload that we have proposed. Data is sent from User1 to User2 after passing through the Load Balancer, it is sent to User3. However, User1 would have the overload caused by increased traffic if sent to User4, it is difficult to ensure the reliability of the data quality of User2 and User3. Therefore, we use the Load Balancer to distribute traffic in the network. Load Balancer to prevent overloading of the User1 when User4 receives data of User1 from User2. The dispersion of the packages can efficiently handle in the traffic via a controller center, which give the existing users send the data quality reliably.

$$T_{max},\ T_{min} \tag{1}$$

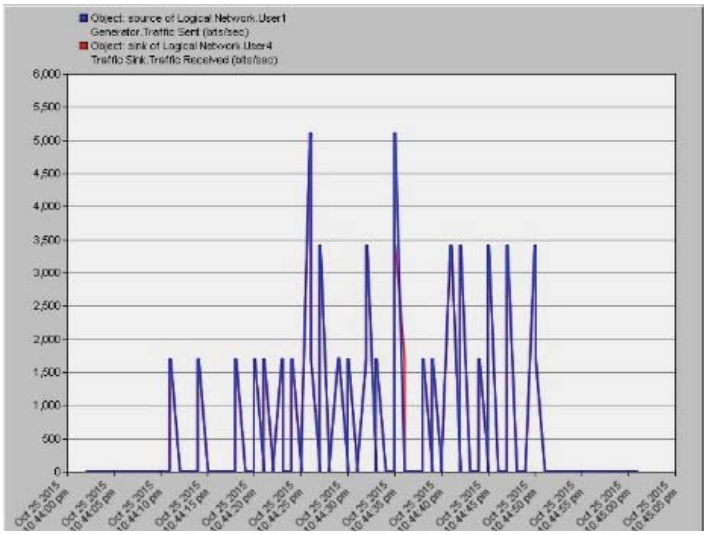
$$T_{total} = \sum_{i=1}^{n} T_i \tag{2}$$

$$T_{avg} = \frac{T_{total}}{n} \tag{3}$$

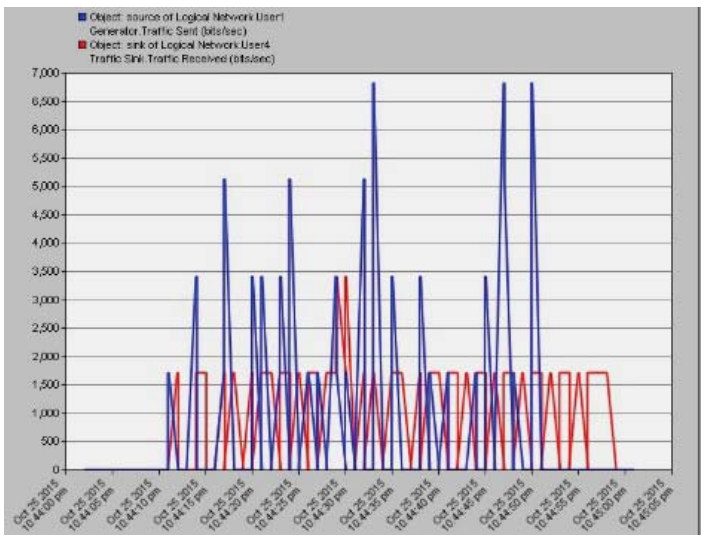
$T_{max}$ of formula (1) is the largest traffic originating from a particular server, $T_{min}$ is the smallest traffic. The $T_{total}$ of formula (2) is a value obtained, the sum of traffic from all the servers, $T_{avg}$ is a value obtained by dividing the sum of the traffic to the total channel of the network. Fig. 3. The processing of the Load Balancer distributes the traffic from the server to generate a server which generates higher traffic than the traffic of $T_{avg}$ and $T_{min}$.

## 4 Experiment

We used the Windows 8.1 desktop-based OPNET Network Simulation Program to evaluate the performance of the proposed system.




(a)




(b)

**Fig. 4** The overload occurs in the normal processing

Fig. 4. X-coordinate of the Y-coordinate is time, and Traffic (bits/sec). Blue line is a server sending data; the Red line is Client (User1) to receive transmitted data.

Fig. 4. (a) is set to the processing speed of 64kbps to set so as to generate and transmit the traffic of the average 1.024bps. As a result, data is received reliably. However, if we have a lot of traffic, (b) shows a picture of an unstable. Fig. 5 shows the results processed by the Load Balancing. Blue line is a server sending data, Red line (User1) and the Green line (User2) is a Client receives transmit data. In the previous figure 4.(b) it had provided unreliable data due to server overload of User1 but in Fig. 5 shows that data is received reliably which is transmitted from the server with User2.

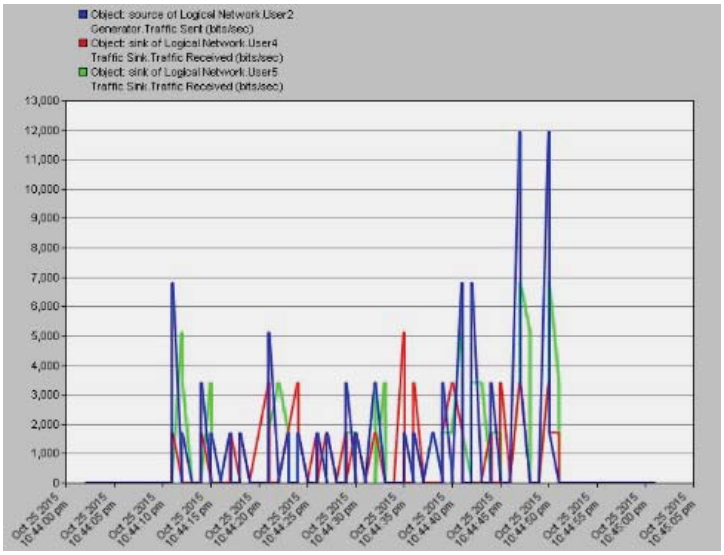



**Fig. 5** Load Balancing Process Result

## 5 Conclusions

In this paper, we propose a Load Balancing method for providing a user a real-time streaming based on WebRTC efficiently. WebRTC is built for real-time streaming environment services which have identified the structure of WebRTC was used by a large number of smart device. In addition, the data communication in real time via a P2P occurs in the process can send and receive large amounts of traffic overloading. The Load Balancing researched to solve the problem, the results of the experiment proved the superiority of the method.

**Acknowledgments** This work was supported by 'The Cross-Ministry Giga KOREA Project' grant from the Ministry of Science, ICT and Future Planning, Korea.

## References

1. Vogt, C., Werner, M.J., Schmidt, T.C.: Leveraging WebRTC for P2P content distribution in web browsers. In: 2013 21st IEEE International Conference on Network Protocols (ICNP), pp. 1–2. IEEE (2013)
2. Granda, J.C., Nuno, P., Suarez, F.J., Garcia, D.F.: Overlay network based on WebRTC for interactive multimedia communications. In: 2015 International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1–5. IEEE (2015)
3. Vidul, A.P., Hari, S., Pranave, K.P., Vysakh, K.J., Archana, K.R.: Telemedicine for emergency care management using WebRTC. In: 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1741–1745. IEEE (2015)
4. Sandholm, T., Magnusson, B., Johnsson, B.A.: On-Demand WebRTC Tunneling in Restricted Networks. arXiv preprint arXiv: 1312.6501 (2013)