An Implementation of P2PTV Service based on WebRTC

Jhon Edisson Villarreal Padilla, 김정호, 이재오

한국기술교육대학교 정보통신공학과

{Villarreal, jungho32, jolee}@koreatech.ac.kr

요 약

WebRTC technology is one of leading various research efforts; this technology allows the exchange of media in a Peer-to -Peer between two browsers, it also provides a number of advantages such as avoidance of 3rd Party plug-in dependency, automatic adaptation to network performance guaranteeing QoS and encryption of communication, but WebRTC was designed to support only browser to browser communication and not multi-browser communication. This creates a challenge in the implementation of multi-browser services such as P2PRadio, Conference and P2PTV. Motivated by this, in this paper the architecture and implementation of P2PTV based on WebRTC are described.

Keywords: WebRTC, P2PTV, HTML5, Javascipt

1. Introduction

Standardization of real-time applications on the web is being led by the Internet Engineering Task Force (IETF) and W3C (World Wide Web Consortium). These two main standardization bodies aim to define the different Application Programming Interfaces APIs to support and enable media sharing in a Peer-to -Peer between two search engines [1]. Within the efforts of the IETF and the W3C, WebRTC was born being based on HTML5 and is composed of PeerConnection, MediaStream and DataChannels. These are APIs that together offer the establishment of a P2P communication. WebRTC brings with them a number of advantages such as avoidance of 3rd Party plug-in dependency, automatic adaptation to network performance guaranteeing QoS and encryption of communication among others. However it was designed to support only browser to browser communication and not multi -browser communication. This creates a challenge in the implementation of services such as Conference, P2PTV, and P2PRadio among others [2].

Furthermore, P2PTV represents an alternative to reduce costs in the conventional IPTV service, as P2PTV, it is based on the distribution of the content used user terminals (in our case the browser) in the role of peers and not as IPTV, which requires a dedicated infrastructure multi-casting [3]. Additionally, P2PTV can scale more easily because each user can become information distributor. This concept has already been deployed between different systems as SopCast [4], Joost [5], among others, however, none of these implementations has used the WebRTC technology.

The advantages of WebRTC and P2PTV technology are used as motivation. This research describes the architecture and implements service P2PTV based on WebRTC service. It also optimizes the location-based service user; the rest of the paper is organized as follows: WebRTC technical concepts and components are summarized in section II and some of the work related to it is also mentioned. In Section III, P2PTV architecture is described in more detail, as well as the description of operation is explained. Section IV is devoted to detailed step by step implementation and results. Finally, the article is concluded, and future works are proposed.

2. Background and Related Works

WebRTC 1.0 API is defined in [6] and enables JavaScript to execute actions that allow the browser to have

real-time communication. This API is mainly divided into three blocks that are the following:

- **PeerConnection:** this allows direct communication between two browsers. However, the communication should be coordinated through signaling protocols such as SIP, REST and others through the Web server (WebSocket or XMLHttpRequest).
- **MediaStream:** transmitted as soon as the PeerConnection is established and is responsible to manage and control everything related to data stream of audio or video. The MediaStream can capture information from any local devices such as webcam or microphone through getUserMedia () function.
- **DataChannel:** contains the transport service, which enables bidirectional communication browser.

Another important aspect is the WebSocket as mentioned in the PeerConnection, which is defined by the IETF in RFC 6455 [7] which allows a bidirectional connection between client and server. The other important protocol is the SIP(Session Initiation Protocol) that is defined in RFC 3261 by the IETF [8] which is responsible for managing all the media sessions(video, voice, file etc.).

Furthermore, as mentioned in the introduction one of the main drawback is that WebRTC was not designed for multi-browser communication whereby the article [9] was relevant to us. In this article, the authors present different models and architecture to support the conferencing service using WebRTC. These are based on a Webserver, It stores the address of the media session and then distribute it to the other participants when a new user is entering to the conference. The WebServer is also responsible for maintaining updated to the other participants in each conference. Based on this we can say that a kind of mesh between all participants is generated. This article was the basis of our architecture we perform but a series of changes that will be mentioned later.

At the time of writing, no research was found regarding the implementation of P2PTV service using WebRTC. However, in the article [10], the author makes a summarized detail of the different architectures of service P2PTV showing the advantages and disadvantages of each one. This research was based on the Tree-based overlay architecture where the original content is divided into different nodes, and each of these nodes is converted into a new media distributor which can be divided into different nodes, and so on. This way, you can optimize the bandwidth of the network. Most important thing for making this architecture efficient is that you should maintain the tree balanced, stable and short. It should also enable a fast and efficient mechanism reconnection when one of the nodes fails. In internet, we found a lot of demos and examples around WebRTC. In [11], there are a wide range of WebRTC application. The most related with our proposal is the Video one-way broadcasting, which supports 256 peer connection, however, it has some drawbacks such as Blurry video experience, Unclear voice and audio lost and Bandwidth issues / slow streaming / CPU overwhelming. The authors said that the solution of those drawbacks is to implement a Media Server. There are also othe interesting services such as screen sharing, text chat, file sharing and so on.

3. P2PTV Architecture based on WebRTC Design

In this section, the architecture of the overall P2PTV service is described, and it is based on WebRTC, adding optimization based on user location. The data flow is also shown between the web server and browsers. Figure 1 shows the overall architecture of P2P service which consists of Streaming Media Server, Web Server, Country Node, Node City and the user equipment (Computers, Tablets, SetUp box etc.). The Streaming Media Server is the one that contains the channels and the content of each one; it is also in charge of coding. The media server streams the required data by the WebServer. The WebRTC video codecs are not yet defined. However, The most implemented protocols are VP8 and H.264 for audio and G.711 and Opus are mandatory, and others are optional. Additionally, the transport Media is Secure Real Time Transport Protocol (SRTP) and the new RTP profiles [12]. The WebServer is responsible for managing the user profile and balancing the shaft for better bandwidth management, plus host (host) website with the user interface.

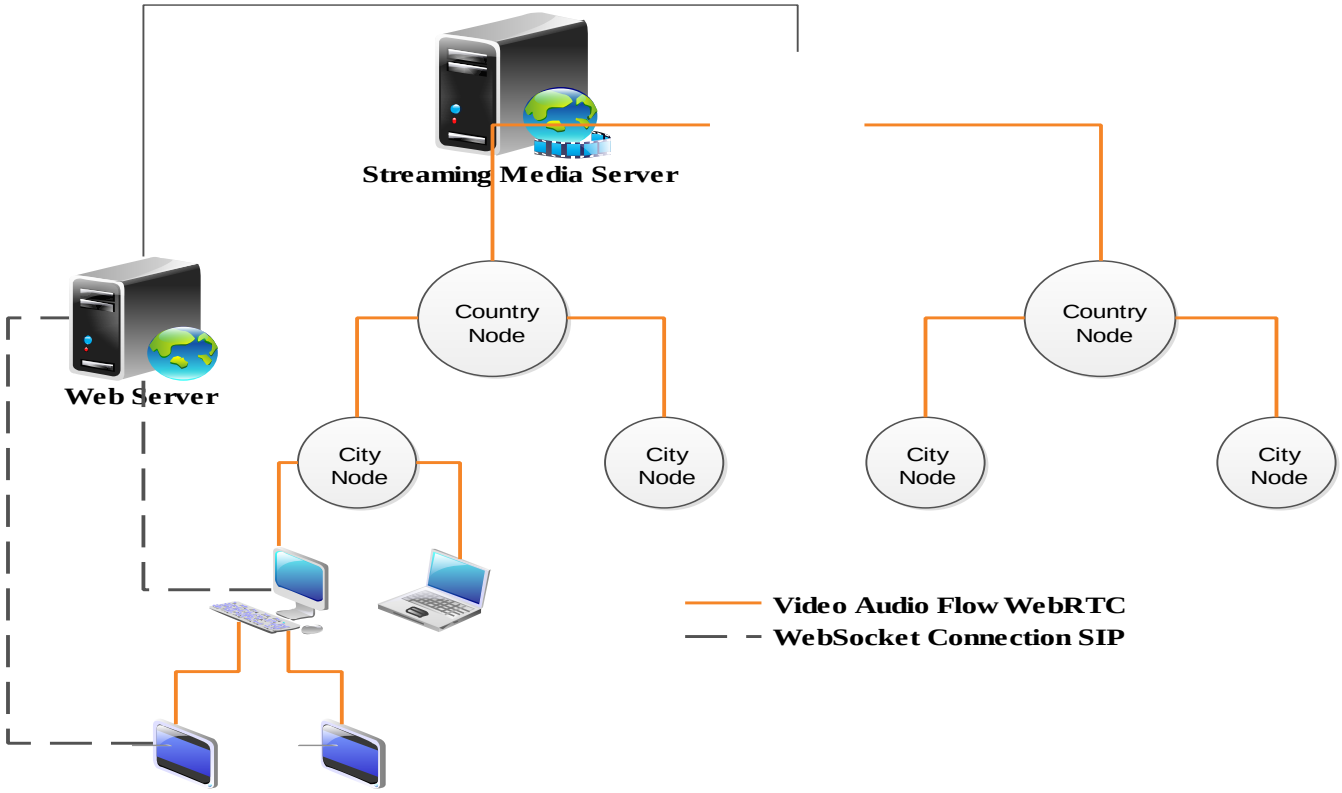

Figure 1. Overall Architecture

The Country Node Streaming Media Server has a smaller capacity than the streaming media server and It is responsible for retransmitting information to the various City Nodes. The City Node are similar but on a smaller scale and is responsible for distributing media to User Equipment.

Finally User Equipment is who receives and displays the required channels in the WebServer, this can also redistribute the information that is getting to different users requiring the same channel.

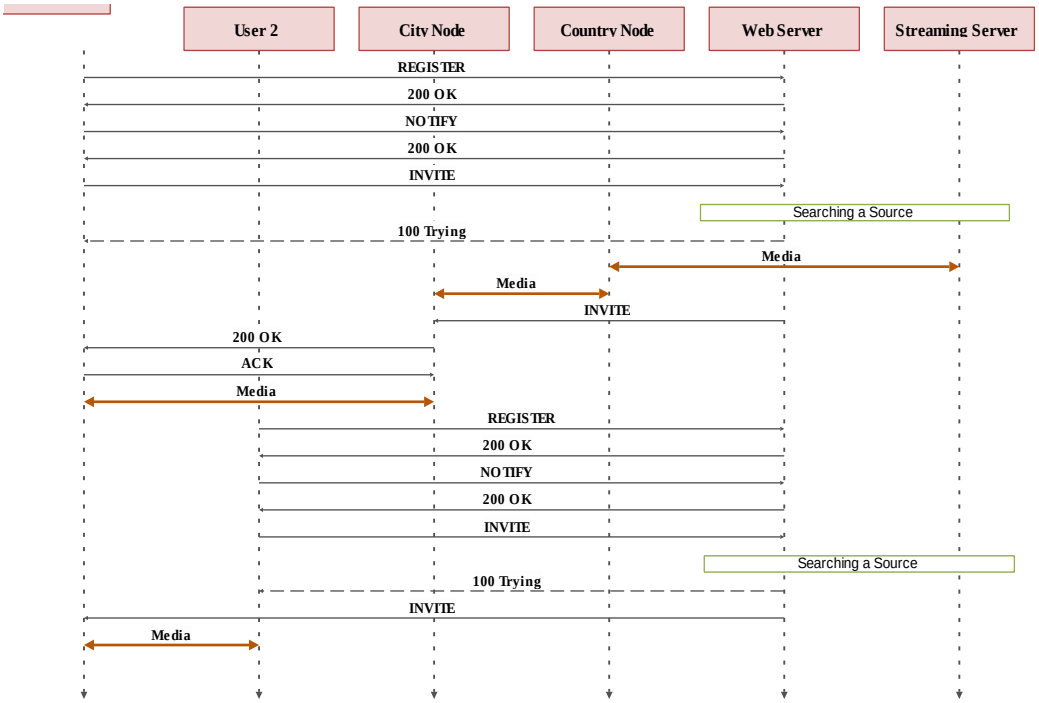

Figure 2. Call Flow

Figure 2 is intent on showing the operation of the above architecture. Let's suppose the first user wants to

apply a channel, the first thing that the user needs to do is register on the WebServer, this is done by sending a SIP REGISTER message which is validated in the WebServer. In order to carry out the authentication and authorization of the user, then the user must register its location WebServer through a SIP PUBLISH message, this information is stored in the WebServer then based on it to perform the tree balance. After the user notifies its location, the user can request a channel for this should send a SIP INVITE message requesting the channel. The WebServer should reroute this message according to the following algorithm:

- The WebServer validates if another user has requested the channel in the same city.
- If so, you must validate if the user has also requested the channel which is enabled to broadcast it, in case if you can redirect the INVITE to that user.
- Otherwise, it will make the same procedure as a country mode.
- If no other user is found to retransmit the information, the WebServer will request Streaming Media Server to forward the information and City Node user and then forward the INVITE of the user to its City Node to start the transmission.

In our case as the first user, the WebServer must inform the Streaming Media Server to transmit the media to the Country Node, City Node to redirect the INVITE message to the user, and through the exchange of SDP message the MediaStream is obtained (WebRTC component) and the transmission of video and audio is set up. Then let's suppose that a second user from the same country and city you want to apply the same channel by which you must first register and report its position as described for the first user. Subsequently you must apply the channel through the WebServer INVITE message using the algorithm described above, and it is known that there is another user who requested the same channel in the same country and city and relaying is enabled. Whereby the INVITE message is retransmitted to the user previously registered, and the exchange is carried, MediaStream eventually sets up Media Session and, so the tree is being created.

In Figure 2, it can also be seen clearly how it will optimize bandwidth because the Media is transmitted once between two nodes, and the transmission among between two users ends on a large scale. In conclusion, at this point it is important to clarify why the choice of the SIP protocol instead of PeerConnection of WebRTC is. The main reason is that the SIP protocol is implemented in different IMS core network as well as different implemented services are based on SIP[11].

4. Implementation and Results

In this section, the implementation of a table test is described and is similar to the architecture shown in section 3. Currently there are still some limitations to the full implementation of the architecture previously exposed yet attempted to replicate a similar architecture without losing the basis of the original architecture.

One limitation is the lack of a Streaming Media Server already developed, so this was replaced by a user interface that captures video through a web camera as shown in Figure 3. At the same time, obtaining video was ruled by media file, as it still has not been implemented, yet in WebRTC is under development. For us, this will be our city node. On the left side we can see the users that are sending information, the number of users is still limited by the browser, for example, in Chrome it only supports up to 20 WebRTC session and Firefox only 7. Each of these sessions occupies a bandwidth which 2M which will be detailed later. For handling SIP messages, we use SIP -JS [13]. Which implements SIP standard in JavaScript.

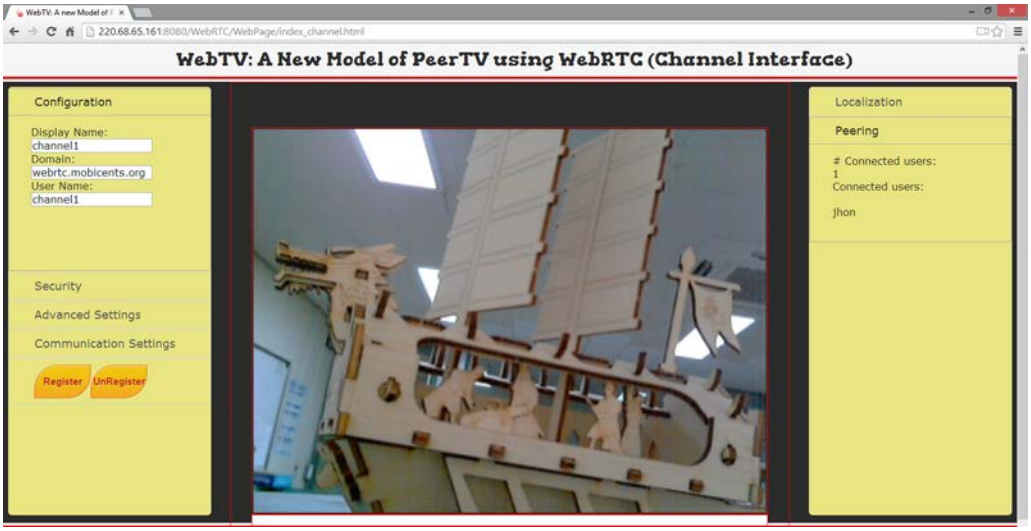

Figure 3. Channel Interface

To implement the Server, we used the Mobicents SIP servlet project [14] which is an open source middleware that has already supported WebRTC and SIP over WebSockets. This project can be installed in any container such as Tomcat or JBOSS. In our case, we use JBOSS which has installed using the eclipse plug-in a computer with Ubuntu 12.04. The algorithm as explained on the location of the new nodes has developed using Java. In Figure 4, it can be observed that the server receives a REGISTER message from the browser and responds with a 200 OK. The website with the user interface is on this server, and the channel is also stored.

```
Call-ID: 1387262826138
CSeq: 1 REGISTER
From: <sip:channel1@webrtc.mobicents.org>;tag=1387262826138
To: <sip:channel1@webrtc.mobicents.org>
Via: SIP/2.0/WS 220.68.65.169:53423;branch=z9hG4bK-333433-f89694c30a5e33124a286129799254be;rport=53423;received=220.68.65.169
Max-Forwards: 70
User-Agent: MobicentsWebRTCPhone/0.0.1
Expires: 3600
Allow: INVITE,ACK,CANCEL,BYE,OPTIONS,MESSAGE
Contact: <sip:channel1@220.68.65.169:53423;transport=ws>
Content-Length: 0

<![CDATA[SIP/2.0 200 OK
To: <sip:channel1@webrtc.mobicents.org>;tag=73349122_06d8ad6a_e3bac888-f8d5-44b5-ab26-3cd13330979f
Via: SIP/2.0/WS 220.68.65.169:53423;branch=z9hG4bK-333433-f89694c30a5e33124a286129799254be;rport=53423;received=220.68.65.169
CSeq: 1 REGISTER
Call-ID: 1387262826138
From: <sip:channel1@webrtc.mobicents.org>;tag=1387262826138
Content-Length: 0
```

Figure 4. User Registration Process

Finally, the user interface using HTML5, CSS and JS-SIP was implemented. It was used for the channel interface for handling SIP messages in a very similar way. In figure 5, different users are displayed requesting the same channel and the information distributed among them.

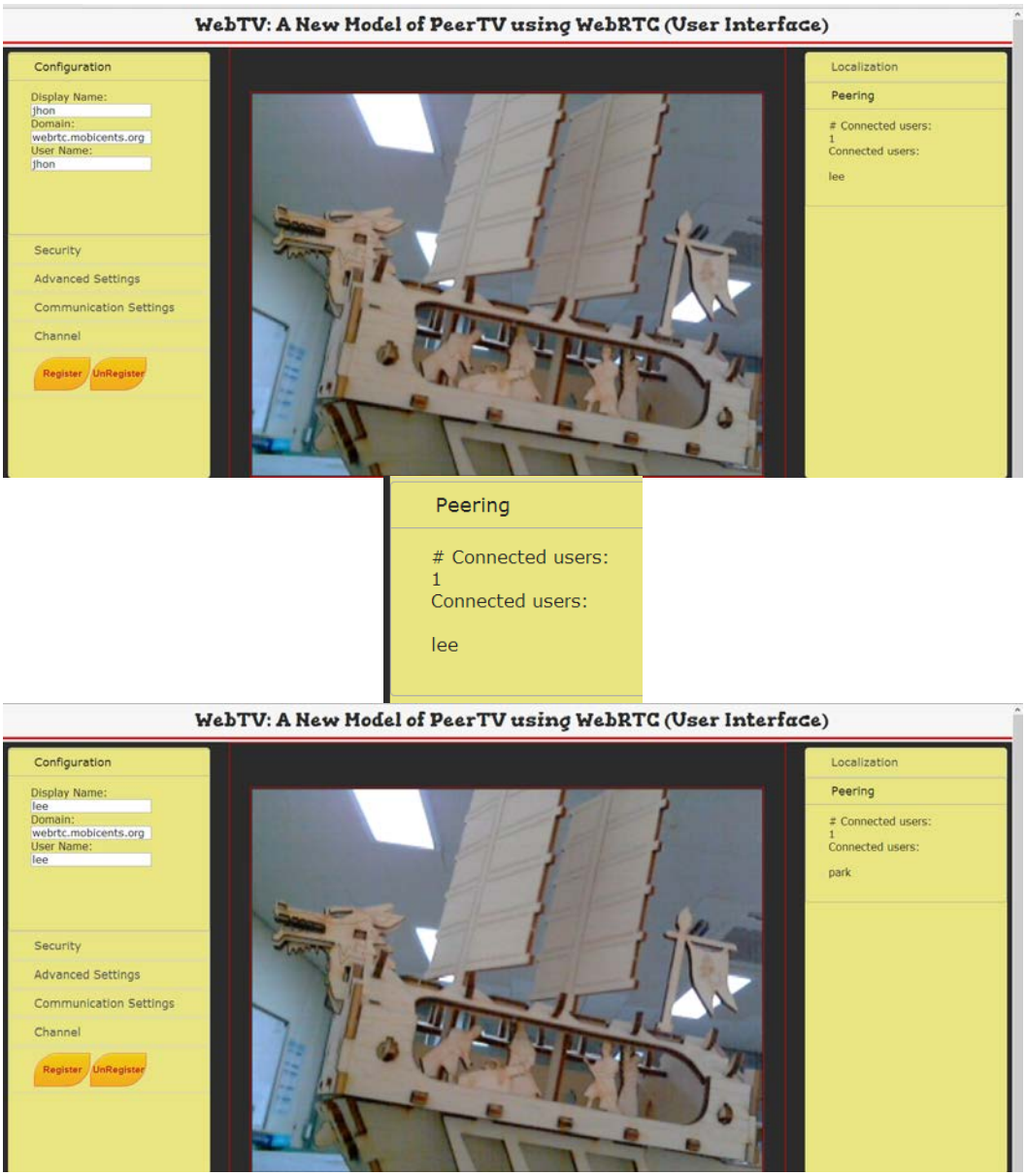

Figure 5. An Example of Users Tree

Figure 5 shows that the user Jhon receives information from Channel 1 and forwards it to the user Lee; the user Lee gets the average user and forwards it to user Park. If it is going on, the size of the tree can be increased.

Furthermore, it is also important to evaluate the bandwidth of the implementation already done, Google Chrome for this in the URL chrome :/ / WebRTC-internals / displays information related to the status of WebRTC session that is established in Figure 6. Statistics of bits sent per second, and packets sent per second on Channel 1 are presented. Where it is only sending information to the user John, so does not exceed 2.0 M in SentPerSecond bit.

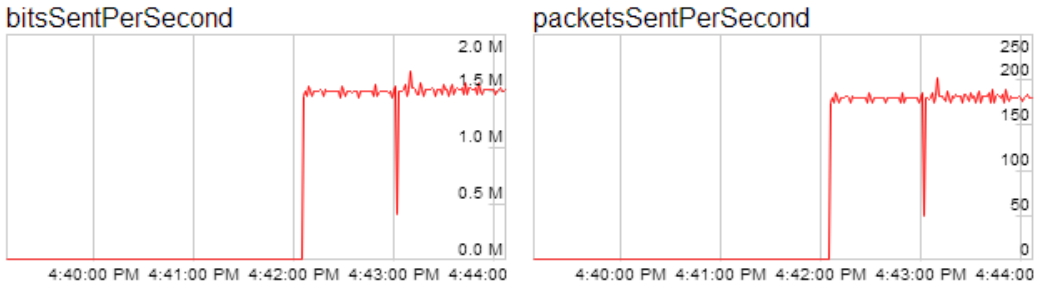

Figure 6. Channel 1 Stats

In Figure 7, it is observed that the user Jhon receives information from Channel 1 and sends information to Lee. In no case exceeding 2M which is the maximum bandwidth that can reserve by default, this value can be changed by modifying the SDP attached the INVITE message. This can be a good example to indicate how the right allocation of bandwidth through the tree is used in the most efficient way.

Figure 7. Jhon User Stats

5. Conclusions and Future Works

In this paper, the implementation of P2PTV using WebRTC is proposed and described. P2PTV is a multi-browser system; however, WebRTC was not designed for this type of services. So, this research shown the way how to implement a multi-browser system (P2PTV). This architecture and implementation was described step by step and the related results were shown. It is important to highlight that this implementation does not use a plugin (Silverlight, Flash), and all is supported by the browser, which is the most important advantage of using WebRTC. The other feature of this implementation is that the tree is optimized depending on user's location. Which efficiently usage bandwidth in each branch of the tree. Additionally it is a fully scalable and low-cost architecture because each device of the user becomes a content distributor.

It should be noted that WebRTC is a technology still in development. However, WebRTC could make a potential change in the future of the communications services. For example, any browser could support any real-time services without need additional plugins. It will reduce the costs of services.

In the future, we would like to analyze the parameters of quality of service (QoS) and Quality of Experience (QoE) in our implementation. Additionally, we will implement our proposal over IP Multimedia Subsystem (IMS), which is a delivery services platform implemented in the telecommunications providers like KT, Telefonica among others. Also, we would like to implement a real environment to analyze the performance of

the location distribution system.

6. References

[1] Loreto, S.; Romano, Simon Pietro, "Real-Time Communications in the Web: Issues, Achievements, and Ongoing Standardization Efforts," Internet Computing, IEEE, vol.16, no.5, pp.68,73, Sept.-Oct. 2012. doi: 10.1109/MIC.2012.115.

[2] Elleuch, Wajdi, "Models for multimedia conference between browsers based on WebRTC," Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on , vol., no., pp.279,284, 7-9 Oct. 2013. doi: 10.1109/WiMOB.2013.6673373.

[3] Alhaisoni, M.; Liotta, A.; Ghanbari, M., "Performance analysis and evaluation of P2PTV streaming behavior," Computers and Communications, 2009. ISCC 2009. IEEE Symposium on , vol., no., pp.89,94, 5-8 July 2009. doi: 10.1109/ISCC.2009.5202402.

[4] http://www.sopcast.com/ Access was on 13th of Dec. 2013.

[5] http://www.joost.com/ Access was on 13th of Dec. 2013.

[6] A. Bergkvist et al., "WebRTC 1.0: RealTime Communication between Browsers,"W3C working draft 09, Feb. 2012; www.w3.org/TR/webrtc/.

[7] I. Fette, A. Melnikov, "RFC 6455: The WebSocket Protocol", IETF, December 2011. http://tools.ietf.org/html/rfc6455

[8] J. Rosenberg et al., "SIP: Session Initiation Protocol,"IETF RFC 3261, June 2002.

[9] Wajdi Elleuch, "Models for Multimedia Conference between Browsers based on WebRTC", 2013 Sixth International Workshop on Selected Topics in Mobile and Wireless Computing.

[10] Yaw-Chung Chen, Shang-Shu Li and Kuan-Teng Chen, "An Efficient Source Allocation Approach for QoS Support in P2PTV Systems", 2010 IEEE 34th Annual Computer Software and Applications Conference.

[11] WebRTC Experiments & Demos https://www.webrtc-experiment.com/

[12] Singh, K.; Krishnaswamy, V., "A case for SIP in Javascript," Communications Magazine, IEEE , vol.51, no.4, pp.28,33, April 2013 doi: 10.1109/MCOM.2013.6495757.

[13] SIP-JS: SIP in JavaScript,http://code.google.com/p/sip-js.

[14] SIP Servlets, https://code.google.com/p/sipservlets/wiki/HTML5WebRTCVideoApplication.

Jhon Edisson Villarreal Padilla
2006 ~ 2010 Sergio Arboleda University, bachelor of Systems and Telecommunications Engineer, Dept. of Engineering.
2011 ~ 2013 Korea Univ. of Technology and Education, master, Dept. of Electrical, Electronics & Communication Engineering

Jung-Ho Kim
2009 Korea Univ. of Technology and Education, bachelor, Dept. of Electrical, Electronics & Communication Engineering
2011 Korea Univ. of Technology and Education, master, Dept. of Electrical, Electronics & Communication Engineering
2011 ~ current Korea Univ. of Technology and Education, doctor course, Dept. of Electrical, Electronics & Communication Engineering

Jae-Oh Lee
1993 Kwangwoon Univ., Ph.D., Computer Networking
1994 ~ 1995 Kolon Information and Communication Research, Team Manager
1995 ~ 2001 Korea Telecom R&D, Senior Member
1999 ~ 2002 WarePlus Co. Korea Telecom Venture Company, Vice President
2002 ~ current Korea Univ. of Technology and Education, Professor