International Journal of Engineering & Technology, 7 (4) (2018) 2118-2121

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.13221
Research paper

Better load balancer design for signaling server in WebRTC communication

Santhosh Kumar B. J [1] *, Nithya Sankar [1], Karthika V G [1]

[1] *Department of Computer Science, Amrita School of Arts and Sciences, Mysuru. Amrita Vishwa Vidyapeetham, India*
**Corresponding author E-mail: santhoshbj50@gmail.com,bj_santosh@asas.mysore.amrita.edu*

Abstract

Most WebRTC applications can communicate through video and audio and other modes. WebRTC allows browser-to-browser multimedia imparting on the web. The proposed system emphasizes over the core architecture and allied technologies of WebRTC server accepting video input and producing output, multimedia transmission, peer-to-peer connection establishment and signaling mechanism. The system makes use of many components and focuses on designing a scalable load balancer architecture for the signaling servers. Currently there is a need of high quality and feasible solution that facilitate communication in the browser but WebRTC makes this possible and scalability feature is compared with open source (Licode's WebRTC server).

Keywords: *Erizo; Nuve; Signaling Server Load Balancer; Signaling Server Scalability; WebRTC (Web Real-Time-Communication).*

1. Introduction

With the speedy evolution of mobile Internet and more and more intelligent mobile devices, the research of multimedia communication systems has become a centre of people's concern. Web multimedia applications based on WebRTC have got more and more attention because of its higher development productivity and ease of use. Web Real-Time Communication (WebRTC) is a collection of protocols, standards and JavaScript APIs. The combination of which enables peer-to-peer audio, video and data sharing between browsers. The standards of WebRTC are at present under joint development by W3C(World Wide Web Consortium) and IETF(Internet Engineering Task Force). Instead of depending on third-party plug-ins or proprietary software, WebRTC turns real-time communication into a standard feature that any web application can leverage through a simple JavaScript API.WebRTC is developed for peer-to-peer communication between end users. The communication is happening between browser of the end users over same network. Over internet, WebRTC using STUN(Session Traversal of User Datagram Protocol[UDP] through Network Address Translation[NATs]), TURN(Traversal Using Relays around NAT) and signaling to communicate between users in different geographical locations. To get an external network address STUN server is used it act as NAT. TURN servers are used to relay traffic if direct connection fails on peer-to-peer communication. Signaling will manage the communication coordination between the end points. It also helps to share error messages, media metadata such as codec settings and codecs, media types and bandwidth. These information used to establish secure connections, connection information such as IP and port between end points. Signaling server acts as the first door of a WebRTC communication.

Licode is an open source WebRTC communications platform. It is based on WebRTC technology and built on top of WebRTC. It provides a rapid development of video conference features based on HTML5. Licode can host video conference rooms on the web and implement streaming, recording and any other real-time-multimedia features. Erizo Controllers are used to manage video conference rooms in Licode. It also does the signaling server tasks. In Licode, room is created by server apps through Nuve API and the users will connect to these rooms through Erizo. Nuve act as load balancer for the signaling communication for Erizo. The scalability of a signaling server is explained using Erizo and Nuve as use cases in this paper.

2. Literature survey

1) Singh K (2013), focused on building a web application for communication. Resources which are structured, and data stored hierarchically in the server are accessed by the end-point through the application logic which is running in the browser. The architecture given enables fully cloud based deployment, on-premise or hybrid of the two. Unlike a single web application which controls the user's social data, this model presented in the paper allows any application to access the authenticated resources of the user that promotes application mix ups. For ex., users contacts are created by one application and are used by another based on the permission granted by the user.
2) Singh K(2013), describes living-content, that creates a private overlay of enterprise social data and interactions on public websites through a browser extension and HTML5. It is enabled for starting collaborations and storing private.

interactions in web pages and addresses issues like lack of adoption, privacy concerns and fragmented collaborations which are seen in enterprise social networks. In a data-centric approach this paper shows scenarios for virtual presence, web annotations, in-

teractions and mix-ups such as it shows the presence of users on linked-in pages or will embed a social wall in corporate directory without any help from any websites. This project enables interaction of new groups which cannot be found in existing social networks.

WebRTC technologies are used to build many real time systems. [3] Holmer S (2013), suggested some methods to prevent the packet loss in the real time environment that make use of WebRTC. Researchers developed some tools that can be used to process the video and it can handle the packet loss in the system but it can't control over the end to end delay caused by the system.

1) Ban, Tae-Hak (2014), addressed the development of a real time video conferencing system using the web browser. This technology is implemented using the concept of hybrid web technology. The researchers developed a hybrid web based real time video conferencing system and it use http or web socket to communicate in all web browsers. we can use this system in a set-top box or smart TV. The application will work only in a web browser applied by html5.
2) Kwok-Fai Ng, Man-Yan Ching (2014), suggested that WebRTC support the browsers for real time communication but in case of multiple participants there come the issues of CPU requirement and bandwidth. So in order to solve this researchers developed a system using P2P-MCU technology. This system offers muti-party video conferencing that support smart phones and PCs. In this approach the browser is integrated with MCU and translate the audio and video in real time. The main disadvantage of the proposed system is that will introduce some delay in the process.
3) Zeidan A(2014), this paper describes WebRTC as an upcoming technology which enables web browsers with real-time communications capabilities such as audio, video and data communications using JavaScript APIs. This paper focuses on new possibilities in telecommunication presenting a solution for inter-operation and migration between SIP-based systems and the emerging standards for WebRTC by an exemplary implementation and enhancement of an open source video conferencing system.
4) Taheri S (2015), describes WebRTC is an HTML5 API will allow browsers to establish a peer-to-peer connection to transfer data and media content through JavaScript APIs. This functionality enables broad range of new applications to emerge. This technology is still under process. Hence, to detect performance bottlenecks of different implementations across operating systems and architectures will help improve it significantly, and a benchmark suite would be helpful to accomplish this task. In this paper, they presented WebRTC Bench, a benchmark which measures WebRTC performance for establishing peer-to-peer media streams and data channels and also the WebRTC media and data communication. This paper presented and discussed performance and evaluation of WebRTC implementation across a range of architectures and operating systems.
5) Jian C (2016), WebRTC use peer-to-peer connection on web and it is used in many applications like multimedia, video conferencing system, chat application etc. Researcher implemented a signaling mechanism in WebRTC based application and it is running well on mobile internet. This system uses http and requires low cost hardware, but its disadvantage is that its works only in smart phones.

3. Signaling server and its role in WEBRTC

The process of connection establishment is also known as signaling and negotiation. To establish a communication between two users, browsers should know where the user is located on the web . The IP address of user's device allows any connected device on the network to send the data directly between the user and browser. The RTC Peer Connection is responsible for transfer of data. As soon as devices know how to find one another over the network, devices start exchanging data about which protocols and codecs each device supports. To communicate with each other, user's need to exchange contact information and the rest will be done by WebRTC. Signaling or negotiation consists of the following steps:

a) Create a list of candidates for peer connection.
b) The first user's browser initiate request to another user's browser for establishing connection.
c) The signaling layer notifies another user about the request. User can accept or decline it.
d) The first user's browser is notified about the response.
e) The first User's browser initiates RTC Peer Connection with other user (if accepted).
f) Both the browser exchange resources and location information through signaling server.
g) The connection succeeds or fails.

4. Limitation of the existing system

As like normal load balancers, WebRTC load balancer consider following matrices to decide the routing:

1) Node availability by check IP/host name of the node.
2) Service availability by check port listening status. Nuve, Licode's signaling load balancer also use above techniques for distribute the traffic among clusters in a scalable WebRTC environment. Nuve designed to deploy on a single server which is responsible for create video conference rooms and help users to connect to this rooms from their browser to the Erizo. If the Nuve node fails to provide it's for some reasons, the new communication request will not be established. So the scalability factor needs to be considered while designing a highly available signaling load balancer.

5. Proposed method

In order to design a scalable architecture for a load balancer component of signaling server, following matrices need to be considered:

1) Signaling node availability using frequent IP/host name of the node and service availability.
2) Signaling server usage status such as CPU usage, Memory consumption, Disk IO status.

Each matrices are discussed in detail below:

a) Signaling node availability using frequent IP/host name of the node and service availability.

Signaling server component has to register itself to the load balancer while initiate the service. Usually this is done while starting the signaling server component. This registration can be done on a persistent storage database or in an in-memory storage database which must be commonly accessible by both load balancer component and signaling server. Signaling server has to update the availability status message on given time interval. This availability status message must contain a timestamp of updated time on the place where the registration details is updated. Load balancer component will run a host reachability test by sending ICMP packets to the signaling server on a particular interval of time. The health status of the signaling server is decided by considering both

host reachability check and the timestamp update done by signaling server.

b) Signaling server usage status such as CPU usage, Memory consumption, Disk IO status.

A server resource availability is basically calculated using CPU usage, available primary memory, and Disk IO rate. CPU idle usage of a server is estimated by the idle usage percentage of that server. This value will be always denoted in percentage. It will be calculated using the utilities such as 'vmstat' or 'top'. Free memory availability factor is calculated based on the ratio between total memory and available free memory.

Disk IO factor can be calculated using the disk IO wait value got from the disk utilities such as 'sysstat' or 'iostat'. This factor can be calculated by subtracting IO wait value from 100.To decide the best node to redirect traffic, We need to consider a performance index value, PIv. PIv can be calculated by multiplying the credit value of each health check matrix with the performance value of the server. Here, we are using CPU idle credit as 4, memory free credit as 4, and disk IO credit as 2. This credit is decided by considering the priority of the matrices affects performance.

Let Sh be the signaling host, Cv is the CPU idle factor in Sh, Mv is the free memory available factor on the Sh, Dv is the IO factor, Cc is the CPU idle credit , Mc is the memory free credit, and Dc is the disk IO credit. Then.

PIv = (Cc * Cv) + (Mc * Mv) + (Dc * Dv) / Cc+Mc+Dc

Signaling server with more PIv will get selected by the load balancer to start sending traffic.

Let the time interval for status update and host check is Ti, minimum health check to satisfy to start sending traffic to the node is Hc, the maximum time difference between each service availability message update is Td, last health check time done by load balancer component is LTl, last service availability message updated by signaling server is STl, registered host in the list is Hn.

In general, following pseudocode is proposed to construct the active signaling server list to serve the signaling request traffic.

For each Ti Health_check_count = 0
If host check for Hn is TRUE Then
Host_alive_flag = TRUE Else
If health_check_countis not 0 health_check_count= health_check_count-1
If STl – LTl< Td and host_alive_flag = TRUE Then
LTl = current_time Ifhealth_check_count<Hc Then
health_check_count = health_check_count + 1 If health_check_count = Hc
Then
If Hn is not exist in active_signaling_list Then
Add Hn with details of LTl to active_signaling_list else
Update Hn with details of LTl in active_signaling_list

Following pseudocode is proposed to remove in-active signaling server from the active list of signaling servers.

For each Ti
If current_time_LTl of Hn> Td Then
Remove Hn from active_signaling_list

Following pseudocode is proposed to calculate performance index of a signaling server.

For each Ti
For Hn in active_signaling_list
Hn_Plv = (Cc * Hn_Cv) + (Mc * Hn_Mv) + (Dc * Hn_Dv) / Cc + Mc + Dc
Update Hn_Plv field of Hn in active_signaling_list

When a request for signaling node comes, the Hn with higher Plv will be returned

1. Experimental result

Table 1: Performance Index of Signaling Server

Cc	4			
Mc	4			
Dc	2			
Test case for host #	Cv	Mv	Dv	PIv
1	30.5	25.6	98.2	42.08
2	82.6	40.5	96.4	68.52
3	60.7	45.4	90.3	60.5
4	27.46	16.2	24.3	22.324

Table 2: Comparison of Existing and Proposed Methods

Test Case	Existing Methods	Proposed Method
Selection of available node	Round robin based	Based on performance index value(PIv)
Registration of Nodes done on load balancer	Manual registration of node done on load balancer	Self registration of node done on A commonly accessible data Storage for signaling Server and load balancer
Scalability	Only vertical scalability is possible	Vertical and horizontal scalability is possible
Node health Check being considered to Select load balancing	No	Yes
Load distribution	Load distribution done in Round robin fashion	Load distribution Done almost Equally by considering resource usage of the node

References

[1] Holmer, S., and M. Shemer. "M. Paniconi," Handling Packet Loss in WebRTC." *Proc. of IEEE International Conference on Image Processing (ICIP 2013)*.Vol. 9. 2013.

[2] Singh, Kundan, and VenkateshKrishnaswamy. "Private overlay of enterprise social data and interactions in the public web context." *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*. IEEE, 2013.

[3] Singh, Kundan, and Venkatesh Krishnaswamy. "Building communicating web applications leveraging endpoints and cloud resource service." *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*.IEEE, 2013. https://doi.org/10.1109/CLOUD.2013.39.

[4] Ban, Tae-Hak, Tae-Beom Cho, and Hoe-Kyung Jung. "Study on the Hybrid Web-based Real-Time Video Communication System." *International Journal of Multimedia and Ubiquitous Engineering* 9.3 (2014): 11-20.

[5] Ng, Kwok-Fai, et al. "A P2P-MCU approach to multi-party video conference with WebRTC." *International Journal of Future Computer and Communication* 3.5 (2014): 319.

[6] Zeidan, Adham, Armin Lehmann, and Ulrich Trick. "WebRTC enabled multimedia conferencing and collaboration.

[7] Solution." *WTC 2014; World Telecommunications Congress 2014; Proceedings of*.VDE, 2014.

[8] Taheri, Sajjad, et al. "WebRTC bench: a benchmark for performance assessment of webRTC implementations." *Embedded Systems for Real-time Multimedia (ESTIMedia), 2015 13th IEEE Symposium on*.IEEE, 2015.

[9] Jian, Cui, and Zhuying Lin. "Research and Implementation of WebRTC Signaling via WebSocket-based for Real-time Multimedia Communications." (2016).

[10] Mani Shankar, S., R. Sandhya, and S. Bhagyashree. "Dynamic load balancing for cloud partition in public cloud model using vista scheduler algorithm." *Journal of Theoretical and Applied Information Technology* 87.2 (2016): 285.

[11] Gokuldev. S and Mr. Radhakrishnan, An Improved Log-based Scheduling and Load Balancing in Computational Grid, International Journal of Applied Engineering Research, Volume 10, Issue 11, ISSN 0973-4562, pp. no 33819-33825.

[12] Berman, Piotr, Moses Charikar, and Marek Karpinski. "On-line load balancing for related machines." *Journal of Algorithms* 35.1 (2000): 108-121. https://doi.org/10.1006/jagm.1999.1070.

[13] Harchol-Balter, Mor, Mark E. Crovella, and Cristina D. Murta. "On choosing a task assignment policy for a distributed server system." *Journal of Parallel and Distributed Computing* 59.2 (1999): 204-228. https://doi.org/10.1006/jpdc.1999.1577.

[14] Westbrook, Jeffery. "Load balancing for response time." *Journal of Algorithms* 35.1 (2000): 1-16. https://doi.org/10.1006/jagm.2000.1074.

[15] Cybenko, George. "Dynamic load balancing for distributed memory multiprocessors." *Journal of parallel and distributed computing* 7.2 (1989): 279-301. https://doi.org/10.1016/0743-7315(89)90021-X