Journal of Digital Contents Society Vol. 17 No. 6 Dec. 2016(pp. 581-592)
https://doi.org/10.9728/dcs.2016.17.6.581

A Dynamical Load Balancing Method for Data Streaming and User Request in WebRTC Environment

Linh Van Ma*, Sanghyun Park, Jong-hyun Jang***, Jaehyung Park****, Jinsul Kim*****

Abstract

WebRTC has quickly grown to be the world's advanced real-time communication in several platforms such as web and mobile. In spite of the advantage, the current technology in WebRTC does not handle a big-streaming efficiently between peers and a large amount request of users on the Signaling server. Therefore, in this paper, we put our work to handle the problem by delivering the flow of data with dynamical load balancing algorithms. We analyze the request source users and direct those streaming requests to a load balancing component. More specifically, the component determines an amount of the requested resource and available resource on the response server, then it delivers streaming data to the requesting user parallel or alternately. To show how the method works, we firstly demonstrate the load-balancing algorithm by using a network simulation tool OPNET, then, we seek to implement the method into an Ubuntu server. In addition, we compare the result of our work and the original implementation of WebRTC, it shows that the method performs efficiently and dynamically than the origin.

Keywords : Big data streaming, Dynamical algorithm, Load balancing, Round robin, WebRTC

WebRTC 환경에 데이터 스트리밍 및 사용자 요청에 따른 동적로드 밸런싱 방법

Linh Van Ma*, 박상현, 장종현***, 박재형****, 김진술*****

요 약

WebRTC는 웹과 모바일과 같이 여러 플랫폼에서 세계 최고 수준의 실시간 커뮤니케이션으로 빠르게 성장했다. WebRTC의 현재 기술은 peer와 시그널링 서버에서 사용자가 요청한 많은 양의 큰 스트리밍을 효율적으로 처리하지 못한다. 따라서, 본 논문에서는 동적로드 밸런싱 알고리즘을 사용하여, 데이터 흐름 전달을 제공함으로써 문제를 처리하는 작업을 수행한다. 또한, 사용자가 요청하는 소스를 분석하고 이러한 스트리밍 요청을 로드 밸런싱 구성 요소에 전달한다. 구체적으로 구성 요소는 요청된 리소스와 사용가능한 리소스의 양을 응답 서버에서 결정한 후 스트리밍 데이터를 요청하는 사용자에게 병렬 또는 교대로 전달한다. 이와 같은 방법을 검증하기 위해 네트워크 시뮬레이션 도구 OPNET을 사용하여 로드 밸런싱 알고리즘을 시연 후 우분투 서버에 적용하여 구현한다. 또한 실험을 통해 도출된 결과와 WebRTC의 구현을 비교하여 제안함으로써 기존 방법보다 효율적이고 동적으로 수행되는 지를 보여준다.

키워드 : 퍼지 의사 결정, 추천 시스템, 소셜 네트워크, 개인 서비스

※ Corresponding Author : Jinsul Kim
Received : December 11, 2016
Revised : December 25, 2016
Accepted : Decembe 31, 2016
*, **, ****, ***** School of Electronics and Computer Engineering, Chonnam National University
email: jsworld@chonnam.ac.kr
*** Electronics and Telecommunications Research Institute
email: jangjh@etri.re.kr
▣ This work was supported by 'The Cross-Ministry Giga KOREA Project' grant from the Ministry of Science, ICT and Future Planning, Rep. of

1. Introduction

In the era of the Internet, creating knowledge and sharing information is the need of the life as well as the progress of humankind. Thus, people are more and more involving to share knowledge that requires an efficiency fundamental platform. Indeed, they continually require the platform to be the same as what the real life does. Hence, peer-to-peer real-time communication [1] emerges to adapt the unending demand of the Internet user.

The Internet applications are creating much of social communities [2]. Through the theoretical, ethical and practical issues of using the internet in many studies, the social is an essential resource for researchers wishing to assess how the latest techniques, tools, and methods in internet-mediated research may support and expand research in their own field. Internet environments offer design principles for the development of innovations; features tested, customizable inquiry projects that students, teachers, and professional developers can enact and refine; and introduces new methods and assessments to investigate the impact of technology on inquiry learning. So far, Internet culture is the virtual spaces and places created by the citizens of the Net and their claims to the hotly contested notion of “virtual community”; the virtual bodies that occupy such spaces; and the desires that animate these bodies.

Hence, network communication has a very vital role in the life. Especially, real-time communication such as Peer-to-peer (P2P) [1] which is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session. Unlike the client/server model, in which the client makes a service request and the server fulfills the request, the P2P network model allows each node to function as both a client and server. Typically, peer-to-peer applications allow users to control many parameters of operation. Moreover, the recent emerging technology, WebRTC [3] has a great advantage since a user can make voice calling, video chat, and P2P file sharing without the need of either internal or external plugins. In [4] authors addressed the application of WebRTC in an enterprise that the enterprise network enforces strict security requirements resulting in very restrictive firewalls and network border elements. On the other hand, WebRTC strives to create an end-to-end secure media path between the two browser instances with little or no interference from any intermediate entity.

Communication technologies are increasingly developing in both application and research field because of unceasing user demands. Recent research described many substantial advancements in the field of data stream processing in WebRTC [5-7]. However, there is an increasing number of emerging applications which requires a real-time processing of very large-volume data streams. The volume of the data streams is such that a single node is not able to process even in a single operator of a query for the entire data stream. They put their research efforts on data streaming which mainly focused on scaling in the number of queries or query operators, but overlooked the scalability issue with respect to the stream volume. Besides, due to the pervasive use of computer systems to collect, communicate, and process data, techniques for computing with large amounts of data that do not fit into traditional databases. Thus, big data has become increasingly important. Essentially, any interesting big data computation requires asymptotically linear time in the size of the

Korea(GK16P0100, Development of Tele Experience Service SW Platform based on Giga Media).

data examined because it examines each and every piece of data. Since such computations require relatively sophisticated systems support such as MapReduce or Hadoop [8].

Suppose the situation that a specific user receives thousands of user requests for getting the resource in WebRTC. In this case, if the user does not have enough computing resources, the system itself is slowed down or even the service does not respond any user request. Especially, the current streaming progress between the user and other peers loses their connection, therefore, it increases transmission cost. Another scenario is that thousands of users want to establish a connection with their desire peers, the request must connect and query on the Signaling server before transmitting data in P2P. If the Signaling server cannot handle the requests, it refuses many requests for establishing the connection. The two above scenarios usually occur when a service has millions of users. Thus, in this paper, we propose a load balancing method for handling the above big request/streaming scenarios. On the side of the clients, we provide a load balancing component in which every request or streaming must pass through a handling center to regulate network traffic. The center determines and compares the available resource and requests resource for delivering data in an appropriate way. On the side of the Signaling server, we also have the handling center of the request before pushing those requests to the server. All of the requests must queue and alternate. In addition, we have a priority in special requests.

In the next section, we describe some basic knowledge in streaming big data, real-time streaming, an advance research of WebRTC and its API, and the related research. In the experimental section, we show our efforts to explain the implementing of our proposed algorithm. Finally, we conclude the research with future research.

2. Related Research

2.1 Data Streaming for Big Data in Network Communication

In network communication, data streaming is the transfer of data at a steady high-speed rate sufficient to support such applications as high-definition television (HDTV) or the continuous backup copying to a storage medium of the data flow within a computer. Data streaming requires some combination of bandwidth sufficiency, and for the real-time human perception of the data, the ability to make sure that enough data is being continuously received without any noticeable time lag.

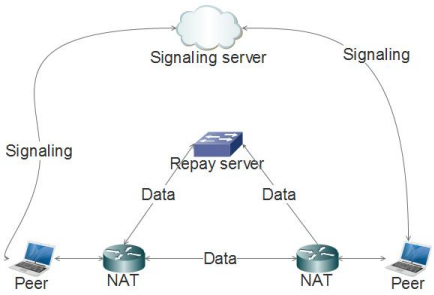
Currently, service providers are looking to leverage the new generation of “Big Data” [9-10] analytics tools and techniques to exploit the untapped reservoirs of data that exist within their network and IT systems. The fine-grained data that can be collected from network probes, intelligent sensors and other data sources are the fuel that powers big data analytics and provides new insights into ways to improve customer experience, optimize networks, drive greater operational efficiency, and enable broader service offerings.

In approaching big data real-time processing, data needs to be quickly processed in nearly real-time to analyze the data to see whether new patterns emerge. In most of the cases, the data is stored in an operational data store, then it is analyzed in the data warehouse by processing those data information in batch and it is not actually in real-time. Therefore, we have no benefit from this operation. Clearly, the analysis has to be fast and practical. Hence, streaming data must focus on speed since mining, data applications require a continuous stream of often

unstructured data to be processed. This requirement is coming more and more into every vertical. Many different frameworks and products are available on the market already, however, we have a small number of mature solutions with good tools and commercial support today. In fact, the mining knowledge of the analysis decreases with time. As such, we must have an efficiency strategy in streaming the data according to network conditions.

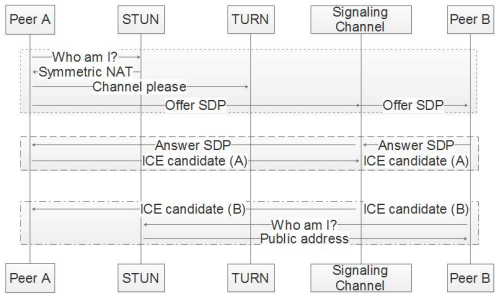
2.2 An Overview of Real-time Communication WebRTC

WebRTC is a recent emerging technology, which provides browsers and mobile applications with real-time communications (RTC) capabilities. The WebRTC components have been optimized to best serve for multiple tasks, but real-time peer-to-peer audio and video communications are the primary benefits. (Figure 1) shows that in order to communicate with another device, a peer uses a signaling process to locate another device, bypasses security and firewall protections, and transmit all multimedia communications in real-time.



(Figure 1) WebRTC signaling process

To avoiding depletion problem of IPv4 addressing space, users now are generally accessed the internet from a work or home-based network. Their computer typically locates behind a firewall and Network Access Translation (NAT), therefore, they do not have a static public IP address. Thus, we need a network information discovery process to connect users by using the Signalling which is based on the Javablocked Session Establishment Protocol (JSEP).



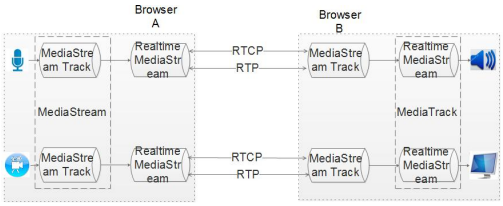
(Figure 2) Establishing connection process in WebRTC

The signaling involves network discovery and NAT traversal, session creation and management, communication security, media capability metadata and coordination, and error handling. Then, the signaling process negotiates and establishes the network session connection between peers. This process is analogous to making a phone call. The initial session negotiation and establishment happen using a signaling/communication protocol specialized in multimedia communications. Any peer that is attempting to communicate with another peer generates a set of ICE candidates, where ICE stands for the Interactive Connectivity Establishment protocol. The candidates are used to represent a given combination of IP address, port, and transport protocol. (Figure 2) depicts the above described process when both Peer A and Peer B establish their connection in WebRTC.

The process of transmitting data between peers depicts in (Figure 3). At the media source, input devices are opened to capture.

Media from the input devices is encoded into packets that are transmitted across the network. At the media destination, the packets are decoded and formed into a media stream. The media stream is sent to output devices. An application that wishes to enable two-way audio and video communications between peers can create four media streams in which an audio stream in each direction, a video stream in each direction.

In addition, WebRTC has data channels which a channel being able to exchange data directly between two browsers, without any sort of intermediary web socket server, is very useful. The data channels carry the same advantages of WebRTC video and audio, it is fully peer-to-peer and encrypted. This means the data channels are useful for things like text chat applications, file transfers, P2P file exchanges, gaming, etc. When the signaling is completed, then we have a peer-to-peer connection between two users which can contain video and audio streams, and the data channel.



(Figure 3) The process of transmitting data between peers

2.3 Load Balancing in Real-time Communication

Distributed systems for big data management very often face the problem of load imbalance among nodes. To address this issue, the load balancing ought to be achieved using an inferred system state; based on locally gathered data or the system must be optimized in structure to allow load balancing to be more easily provisioned. In [11] authors described three potentially viable methods for load balancing in large scale cloud systems. Firstly, a nature-inspired algorithm may be used for self-organization, achieving global load balancing via local server actions. Secondly, self-organization can be engineered based on random sampling of the system domain, giving a balanced load across all system nodes. Thirdly the system can be restructured to optimize the job assignment at the servers. It aims to provide an evaluation and comparative study of these approaches.

Currently, in cloud computing [12], it mainly considers the current system condition in VM (Virtual Machine) resources scheduling, but seldom considers the previous condition before scheduling and the influence on system load after scheduling which usually leads to load imbalance. Therefore, in [13] authors presented a scheduling strategy to realize load balancing. According to historical data and current state and through genetic algorithm, this method computes in advance the influence it will have when the current VM service resources that need deploying are arranged to every physical node, then it chooses the deployment that will have the least influence on the system. In this way, the method realizes the best load balancing and reduces or avoids the dynamic migration.

As more applications are becoming data-intensive and experiencing data explosion [14] such that tasks are dependent and task execution involves processing a large amount of data, data-aware scheduling and load balancing are two indispensable yet orthogonal needs. Migrating tasks randomly through work stealing would compromise the data-locality and incur significant data-transferring overhead. Furthermore, the mapping process may cause poor load balancing due to the potential unbalanced data distribution. Therefore, in the work [15] authors investigated a data-aware work stealing

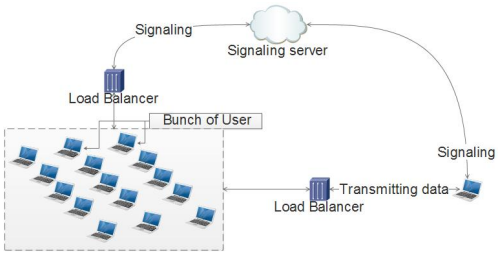
technique that is able to achieve good load balancing, and yet still tries to best exploit data locality. The results showed that their technique is scalable to achieve both good load balancing and high location-hit rate.

The scale of a cloud computing platform would be very large and provides various kinds of operation systems for the consumer, and applications running on the cloud are numerous. Such a cloud environment can be treated as a complex system which has the characteristic of dynamic, open, non-linear interact and a lot of components. Load balancing in the cloud should be distributed, flexible and extensible. Based on [16], authors of [17] proposal a load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. Their results showed that it improves many aspects of the related Ant Colony algorithms which are proposed to realize load balancing in distributed systems.

As compared to P2P file sharing system, the real-time playback constraint of live media poses challenges in designing efficient live streaming systems. The authors in [18] focused on designing request-peer selection algorithms. The simplest approach is to pick up a potential provider randomly. Through extensive simulations, we showed that our algorithm distributes the traffic load more evenly among peers. As a result, the peers' uplink bandwidth is better utilized and the streaming server load is reduced, meanwhile, the quality of experience is also improved.

One way to perform such updates is to design specific dynamic algorithms or incremental algorithms for each specific application of interest. While prior work on self-adjusting computation required to make reasonably substantial changes to the program code, recent developments allow the types of a program to be annotated with just a few keywords and employ a type-directed translation algorithm to generate the necessary changes to the code automatically. Thus, from [19] authors restricted themselves to the MapReduce paradigm. They implemented a single-node version of MapReduce in the implicit self-adjusting language. It allows them to read data from a file system into the memory and change the data incrementally while performing automatic incremental updates after each change.

3. Load Balancing System Proposed Overview

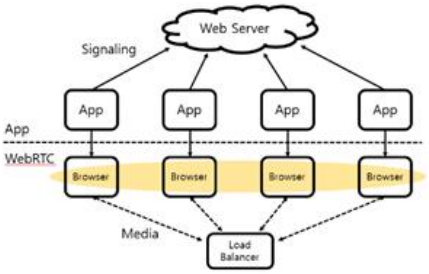


(Figure 4) System proposed overview

In the system, a bunch of users has thousands of users is preparing to send a request to establish a connection to the Signaling server for obtaining resources such as sharing data or video from a specific user as depicted in (Figure 4). In this case, we set up a load-balancer to handle an overloading problem if it occurs. When the connection is ready to transmit data in real-time streaming service, the user uses WebRTC to send and receive video, audio, and data in both directions via the load-balancer. In the load-balancer, we implement our proposed algorithm for delivering data as well as handle the users' requests efficiency. It detects an amount of available resource and requests resource to transmit data appropriately. If the user can handle a certain amount of request, the load-balancer eliminate itself in the transmitting process dynamically.

3.1 System Design

(Figure 5) depicts the overall system architecture for a real-time big-streaming services using WebRTC and a load-balancer. First, a user starts their real-time streaming services, as such, other users can connect to the user via applications which support WebRTC real-time communication such as the Chrome, Firefox browser, and mobile applications. In the process of establishing the connection, the Signaling assists to establish the connection. In which, the Signaling processor in a web-server is a coordinated communication to send and receive streaming data between users. After the process in the Signaling is ended, users send and receive data in real-time over the load-balancer.

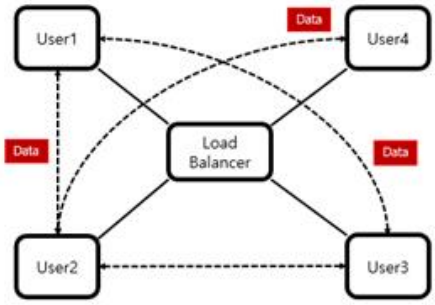


(Figure 5) System Design with WebRTC and Load-balancers

3.2 A Load Balancing Method for Efficient Data Transmission

In a case of large amount of traffic which generated from a big bunch of users, we apply a load-balancing technique to improve the quality of service and reduce the overload of the network system.

(Figure 6) depicts the load-balancing flow and the method to reduce the overload traffic problem that we have proposed. Data is transmitted from User1 to both User2 and User3 after passing through the load-balancer. However, User1 would have the overload of increased traffic if User4 required for obtaining data from User1, thus, it is difficult to ensure the reliability data transmission between User1 and both User2, User3. Therefore, we use the load-balancer to distribute traffic in the network. The load-balancer prevents overloading of User1 by redirecting the requesting data in User4 to receive data from User2 or User3, where data have already transmitted to these users.



(Figure 6) Load-balancing flow to handle big-streaming

The dispersion of the packages can efficiently handle the traffic via a controller center, which assists the existing users to send the data reliably. Hence, the amount of data distribution is expressed as follows equations,

$$T_{\min} \ll T_{\max} \quad (1)$$

$$T_{total} = \sum_{i=1}^{n} T_i \quad (2)$$

$$T_{avg} = \frac{T_{total}}{n} \quad (3)$$

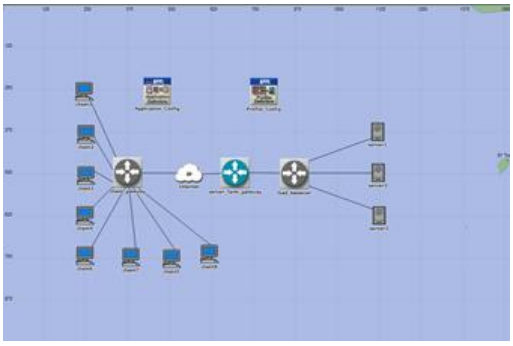
$T_{\max}$ in formula (1) is the maximum traffic which a particular server can handle to transmit, $T_{\min}$ is the minimum traffic data

from the server. The T_{total} of formula (2) is a value obtained by summing 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.

4. Simulation and Experiment

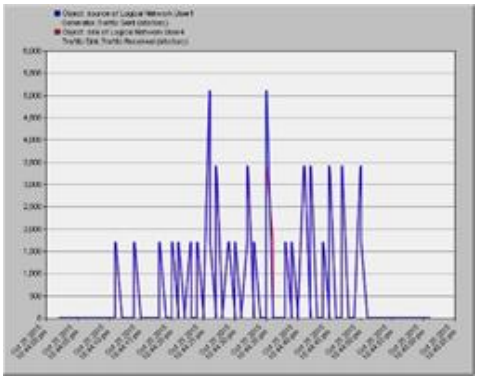
4.1 Simulating the Proposed Load Balancing with OPNET

In this section, we conduct the experiment in Windows 7 desktop-based OPNET [20] network simulation program, which is a software that provided performance management for computer networks and applications, to evaluate the performance of the proposed load-balancing algorithm. The system simulation is depicted in (Figure 7) with eight clients on the left, and three streaming servers which support real-time communication on the right. The load-balancer in the middle of the connection has responsibility for distributing data if eight users focus on achieving data on a specific server.

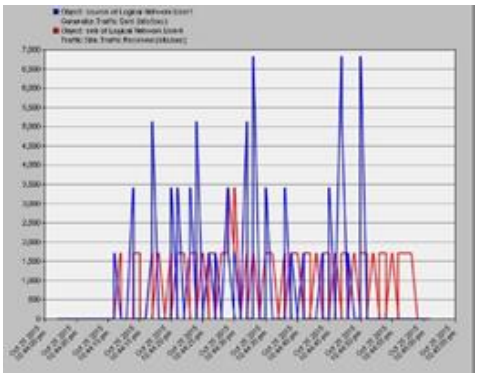


(Figure 7) System infrastructure simulation overview

Both (Figure 8-a), (Figure 8-b) depict that X-coordinate is the transmitting time and Y-coordinate is the amount of traffic transmission in bits/second. The blue line is the data which is sent from the server; the red line is the received data in the client (User1).



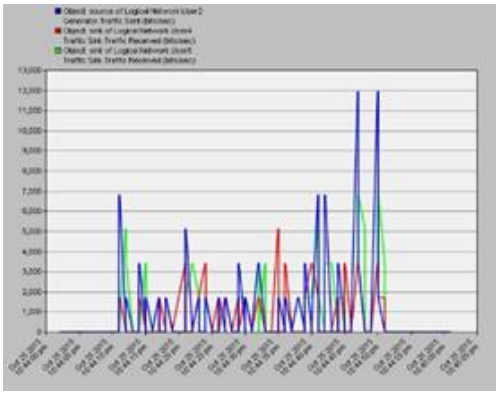
(Figure 8-a) The overload occurs in the normal processing



(Figure 8-b) The overload occurs in the unstable network traffic

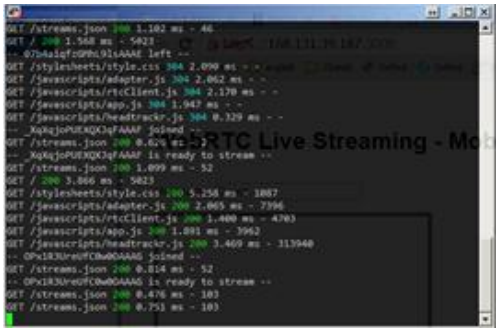
In addition, in (Figure 8-a) the transmission rate is set to the processing speed of 64kbps in each user so that the average generated transmit is 1.024bps. As a result, data is received reliably. However, if we have a lot of traffic, (Figure 8-b) shows a picture of an unstable traffic. Furthermore, (Figure 9) shows the results processed by the load-balancing. The blue line is a server sending data, the red line (User1) and the green line (User2) are graphs of receiving data from two clients. In (Figure 8-b), the servers had provided

unreliable data due to the server overloading data, but in (Figure 9) shows that data is received stably which is transmitted from the servers.



(Figure 9) Load balancing result of delivering data on the three servers

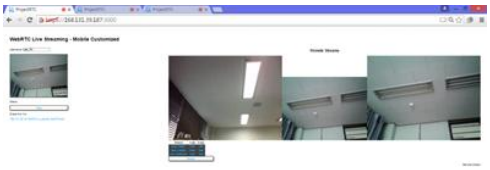
4.2 Implementation of the Proposed Method in Ubuntu



(Figure 10) Handling command in the Signaling server

We have installed the WebRTC Signal server in Ubuntu 12.04 by using Node.js [21]. In the system, we implemented our load-balancing algorithm. As shown in (Figure 10), the server handles requests from browser user and displays user's request as well as responding time.

(Figure 11) shows the real-time streaming services through WebRTC. We have four devices which are Tablet, Notebook, PC, and Phone. All of them have established the connection with the Signaling server, *https://168.131.39.187:3000/*. Each of these devices sent and received a video and voice stream in real time via the P2P communication using the Chrome web browser. Then we access the WebRTC monitoring plug-in in Chrome which was provided by Google at *chrome://webrtc-internals/* to observe data transmission variation. (Figure 12) shows that the transmission rate is steady without any corruption of the big-streaming process.



(Figure 11) Implementation of WebRTC load-balancing algorithm in Ubuntu

(Figure 12) A statistic of package sending and receiving in video channel

5. Conclusion

In this paper, we have proposed the load-balancing method for improving WebRTC-based real-time streaming service efficiency when a lot of users establish a connection for getting data. It causes the

overloading problem. To show how the method works and its performance, we set up the simulation in the network simulation tool OPNET, it showed that after several seconds the transmission is balanced appropriately to the available resource of the user who broadcast their data to a bunch of large users. In addition, we also implemented the method in WebRTC original API, four devices use to access the Signaling server, and the monitoring result showed that the client and the server handle the large requests, efficiency without any corruption at both service provider and client which receives streaming data. In the future work, we will focus to improve the algorithm with a smart system, which combines with CND or central distributing server to reducing traffic generating from users.

References

[1] A. Passarella, "A survey on content-centric technologies for the current Internet: CDN and P2P solutions," Computer Communications, Vol.35, pp.1-32, 2012.

[2] M. C. Linn, "Internet environments for science education: Routledge," 2013.

[3] E. Rescorla, "WebRTC Security Architecture," 2015.

[4] A. Johnston, J. Yoakum, and K. Singh, "Taking on WebRTC in an enterprise," Communications Magazine, IEEE, Vol.51, pp.48-54, 2013.

[5] Lee HN, Kim DH, "Selection of Scalable Video Coding Layer Considering the Required Peak Signal to Noise Ratio and Amount of Received Video Data in Wireless Networks," Journal of Digital Contents Society, Vol.17, No.2, pp.89-96, 2016.

[6] Linh. M. Van, J. Kim, S. Park, J. Kim, and J. Jang, "An efficient Session_Weight load balancing and scheduling methodology for high-quality telehealth care service based on WebRTC," The Journal of Super computing, Vol.72, No.10, pp.3909-3926, 2016.

[7] Linh. M. Van, Jang JH, Kim J, "Adjusting Local Network Speed by Using Fuzzy Theory with An Illustration in WebRTC Environment," Journal of Digital Contents Society, Vol.16, No.6, pp.917-25, 2015.

[8] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, Vol.51, No.1, pp.107-113, 2008.

[9] P. Zikopoulos, C. Eaton, and others, "Understanding big data: Analytics for enterprise class hadoop and streaming data," McGraw-Hill Osborne Media, 2011.

[10] Kong HS, Song EJ, "A Study on Hotel Customer Reputation Analysis based on Big Data," Journal of Digital Contents Society, Vol.15, No.2, pp.219-25, 2014.

[11] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference, pp.551-556, 2010.

[12] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[13] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual ma-chine resources in cloud computing environment," in Parallel Architectures, Algorithms and Programming (PAAP) Third International Symposium, pp.89-96, 2010..

[14] A. S. Szalay, G. Bell, J. Vandenberg, A. Wonders, R. Burns, D. Fay, et al., "Graywulf: Scalable clustered architecture for data intensive computing," in System Sciences, HICSS'09. 42nd Hawaii International Conference, pp.1-10, 2009.

[15] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang, and I. Raicu, "Optimizing load balancing and data-localit

y with data-aware scheduling," Big Data 2014 IEEE International Conference, pp.119-128, 2014.

[16] Z. Zhang and X. Zhang, "Realization of open cloud computing federation based on mobile agent," Intelli gent Computing and Intelligent Systems ICIS 2009 IEEE International Conference, pp.642-646, 2009.

[17] Z. Zhang and X. Zhang, "A load balancing mechanis m based on ant colony and complex network theory in open cloud computing federation," in Industrial Mechatronics and Automation (ICIMA), 2010 2nd In ternational Conference, pp.240-243, 2010.

[18] N. Liu, Z. Wen, K. L. Yeung, and Z. Lei, "Request-p eer selection for load-balancing in P2P live streamin g systems," in Wireless Communications and Netw orking Conference (WCNC) IEEE, pp.3227-3232, 20 12.

[19] U. A. Acar and Y. Chen, "Streaming big data with self-adjusting computation," Proceedings of the 201 3 workshop on Data driven functional programming, pp.15-18, 2013.

[20] O. Modeler, "OPNET Technologies Inc," 2009.

[21] M. Cantelon, M. Harter, T. Holowaychuk, and N. Rajlich, "Node. js in Action: Manning," 2014.

Linh Van Ma

2013 : Hanoi University of Science and Technology, Vietnam (B.S. Degree).

2013～2015: Researcher and Developer at Samsung Vietnam Mobile R&D Center - SVMC.

2015～now : School of Electronics and Computer Engineering - Smart & Media Computing Laboratory.

Research Interests : Mobile Cloud Computing, Next Generation of Mobile Platform, Mobile Operating System and Peer-toPeer Network

Sanghyun Park

2010 : Computer and Information from the University of Korea Nazarene.

2012 : Chonnam National University, School of Electronic and Computer Engineering
(M.S Degree)

2010～2012 : Engineer in System Development Team of Media Flow Company.

2014～now : Doctoral candidate in Electronics and Computer Engineering at Chonnam National University.

Research Interests : Interactive Media, Systems Development, Embedded of systems, Digital Media and Cloud computing.

Jong-Hyun Jang

1988 : Kyungpook National University of Electronic Engineering(B.S Degree).

2000 : Chungnam National University of Computer Science (M.S Degree)

2004 : Hankuk University of Foreign Studies of Information and Communication (Ph.D Degree).

1988~1994: Assistant Manager in Daewoo Telecom Co., Ltd.

1994~now: Researcher in Real and Emotional Sense Platform Research Section, Electronics and Telecommunications Research Institute (ETRI).

Research Interests : Real-time Middleware for Telecommunication Systems, Home Networking Systems, and Real-Sense Media Services

Jaehyung Park

1991 : Yonsei University (B.S. Degree).

1993 : Korea Advanced Institute of Science and Technology (KAIST) (M.S Degree).

1997 : Korea Advanced Institute of Science and Technology (KAIST) (Ph.D Degree).

1997~1998: Researcher in Center for Artificial Intelligence Research(CAIR), KAIST.

1998~2002: Senior Researcher in Network Laboratory, Electronics and Telecommunications Research Institute (ETRI).

2002~now : Professor in Chonnam National University, Gwangju, Korea.

Research Interests : Internet Protocols, Multicast Routing, Wireless Mesh/Ad-hoc Networks, and Network Security/Cryptography.

Jinsul Kim

2001 : University of Utah, Salt Lake City, Utah, USA (B.S. Degree).

2005 : Korea Advanced Institute of Science and Technology (KAIST) (M.S Degree).

2008 : Korea Advanced Institute of Science and Technology (KAIST) (Ph.D Degree).

2005~2008: Researcher in IPTV Infrastructure Technology Research Laboratory, Broadcasting/Telecommunications Convergence Research Division, Electronics and Telecommunications Research Institute (ETRI).

2009~2011: Professor in Korea Nazarene University, Chon-an, Korea.

2011~now: Professor in Chonnam National University, Gwangju, Korea.

Research Interests : QoS/QoE, Measurement/ Management, IPTV, Mobile IPTV, Smart TV, Multimedia Communication and Digital Media Arts.