A Novel NAT-based Approach for Resource Load Balancing in Fog Computing Architecture

Chin-Feng Lai[1], Hung-Yen Weng[1], Hao-Yu Chou[2], Yueh-Min Huang[1]

[1] Department of Engineering Science, National Cheng Kung University, Taiwan
[2] Sercomm Corporation, Taiwan
cinfon@ieee.org, hungyenweng@google.com, jimmy19911108@hotmail.com, huang@mail.ncku.edu.tw

Abstract

In recent years, with the rise of the internet of things technology, most devices can be connected to the internet, resulting in fewer and fewer IP addresses. Therefore, the number of applications using NAT has also increased. Coupled with the rise of fog computing architecture, NAT traversal has become increasingly difficult to be implemented. Although many methods of traversing firewalls have been proposed and widely used in various communication archiectures, Users cannot rely solely on the central server to establish connections in the peer-to-peer network, which increases the loading on the NAT architecture and transport server. In the past, there have been related papers to solve this problem, and the goal is to increase the success rate of establishing a connection through a NAT server. Therefore, a novel load balancing solution is proposed in this study, and the loading value obtained from the SVM model is adopted as the basis for selecting the network address conversion server. At the end of the study, we not only discuss the maximum server loading, different analysis models, and the delay time added by the new processes in the architecture, but also find the proposed approach is able to achieve the loading balance with only a small increase in delay time.

Keywords: NAT-based, Resource load balancing, Fog computing architecture

1 Introduction

Peer-to-peer networking has been widely applied since it was proposed in the late 1960s. Its application fields include file sharing, multimedia streaming, network communications, etc., but Network Address Translation (NAT) was proposed in the mid-1990s Later, it caused difficulties in the connection between the two points in the peer-to-peer network. The original development purpose of NAT was to solve the problem of insufficient Internet Protocol version 4 (IPv4) addresses. Later, NAT was combined with firewall technology to develop Provides private network protection methods. Although these methods can improve the security of private networks, they also cause peer-to-peer network connections. Endpoints outside the firewall cannot know the IPv4 address of the internal end point. To establish a peer-to-peer network connection, Interactive Connectivity Establishment (ICE) technology was proposed to solve this problem.

The ICE technology was developed by the Internet Engineering Task Force (IETF) in 2003 and was published as RFC5245 [1]. This technology uses Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN). NAT traversal is achieved through the road protocol. The above two protocols are also listed in the IETF standard documents, namely RFC5389 [2] and RFC5766 [3]. After STUN assists both ends of the peer-to-peer network to obtain the information required by both parties to establish a connection. It will no longer intervene in the connection, but not all firewalls can be traversed by STUN. When the traversal fails, the TURN server must be used as a packet stream relay, so that ICE can be used to establish a peer-to-peer network connection. However, with the rise of network security awareness, the need to use the TURN server to establish a connection is also increasing.

At present, many network communications provide services through peer-to-peer connections in daily life, and these service providers are also facing the problem of increasing usage of relay servers that implement the TURN protocol. There are related documents in the past proposed solutions [4-6], hoping to avoid the use of relay servers when connecting. However, although the usage of relay servers can be effectively reduced under certain circumstances, there are still some differences in the NAT mechanism of network products on the market, resulting in the effect is limited. In view of the above factors, a load balancing method is proposed for the relay server in this study, and a new system architecture is designed in conjunction with the ICE process, hoping to find another solution to this problem to reduce the relay server loading.

Therefore, this paper has two contributions:

*Corresponding Author: Yueh-Min Huang; E-mail: huang@mail.ncku.edu.tw
DOI: 10.3966/160792642021052203002

(1) Design a new system architecture and method based on the ICE process.

To propose a solution different from Skype, we will first understand the Skype connection mechanism through relevant research before designing the new system architecture, and then learn the Skype system architecture, and at the same time, achieve the TURN server under the condition of maintaining the ICE firewall traversal process the purpose of load balancing.

(2) Load balancing performance evaluation of the new architecture.

Under the proposed framework, we simulate users to conduct instant communication in a peer-to-peer network environment, and capture the loading caused by this method in different situations. After analysis, we know the performance of the proposed method and framework for load balancing. which performed.

The remainder of this paper is organized as follows. Section 2 discusses related works and results; Section 3 introduces the NAT-based aapproach for resource load balancing. Section 4 evaluates the approach performance, and test the implementation system; finally, Section 6 gives conclusions.

2 Related Works

2.1 Interactive Connectivity Establishment

ICE [1] has been widely used after being published by IETF, including SIP [7], WebRTC developed by Google [8], and many peer-to-peer network products and multimedia streaming related research on the market. ICE has its own authentication method in terms of security, and no matter whether the peer-to-peer connection is established successfully or not, there is its connection method. According to the description in [1] and the collation of [9], the connection establishment process of ICE can be divided into The seven steps are collection, prioritization, coding, Offering and Answering, checking, and completion in order. When a host inside the NAT wants to establish a connection to the outside, it will first collect the necessary IP and port, which is called candidate in ICE, and Use STUN to obtain the external information of one's own NAT, then establish a priority order for the collected candidates for future selection, and then use the collected candidates to construct the connection and stream encoding information, and use the Signaling Server Send a connection request to the other party by means of offer and answer, and complete the information exchange at the same time. After the information exchange is completed, ICE will check whether the connection can be established. When the check is completed, the two parties will start multimedia streaming, but according to NAT type, the way to successfully establish a connection with STUN is limited to less rigorous types than symmetric NAT. And whether the port-limited cone NAT can be used, it depends on the implementation of the peer-to-peer connection system depending on the method, when the connection cannot be established using STUN, it will switch to the TURN server. TURN is an extension of STUN. It will forward the multimedia stream packets of both parties and become a relay server. The information format in the agreement is the same as STUN. The relevant content can refer to the RFC document. Before ICE was proposed, many NAT traversal solutions have been proposed, including static routing, application gateway (ALG) [10], session border controller (SBC) [11], general-purpose accessory Universal Plug and Play (UPnP) [12] and so on. Among them, static routing is for users to set NAT rules by themselves; ALG will change application layer information on the NAT device; SBC uses the server as a proxy to packets are forwarded by relay; UPnP changes the routing table on the NAT device to achieve the purpose of NAT traversal. Although the above methods can achieve NAT traversal, it is difficult of setting, NAT traversal capability, and device support restrictions due to network security. Which makes the above methods not widely accepted. ALG, SBC and ICE do not require users to set additional NAT devices. Among the three, SBC and ICE do not need NAT equipment to have additional support in function, and of the two, ICE can use STUN to reach P2P connections, and is more complete than SBC in terms of security mechanism, compared with other NAT traversal methods Together, it has the best performance in terms of user friendliness, device support, NAT traversal capabilities, and protocol security.

2.2 Resource Load Balancing

Resource load balancing is an important issue in the fields of web servers, databases, or cloud computing [13-16]. The resources that cause loading issues can be memory, hard disk, central processing unit, or network frequency. Forgiveness, when the server provides services to users, that is, all users share server resources. If the loading is all concentrated on a single server, it may cause problems such as system performance degradation and service quality degradation. At this time, load balancing is It is a great way to solve the problem.

According to the description in [17], the goal of load balancing is to improve system performance, maintain system stability, establish fault-tolerant systems, and adapt to future modifications. In addition, reference [18] also mentioned that load balancing algorithms need to pay attention to the problems include the distribution of nodes in the system, the storage and backup methods of data in the nodes, the complexity of the algorithm, and the impact of a single node failure on the system. The above points will affect the system delay, storage resource utilization and fault tolerance.

Load balancing algorithms can be roughly divided

into static load balancing and dynamic load balancing. The introduction of related algorithms can be seen in [17-20]. Static load balancing will be performed before the system runs. Obtain the available resources of the nodes in advance to allocate work, and because the node resources are not monitored in real time while the system is running, when the loading on the node fluctuates, the previous work cannot be redistributed, and the system performance may be affected. Dynamic load balancing It will obtain available resources in real-time monitoring and allocate work, which can solve the shortcomings of static load balancing. However, because dynamic load balancing requires communication between nodes, the response time will be relatively long.

In addition to the above two types of load balancing methods, later studies have also proposed hybrid load balancing. As introduced in [20], hybrid load balancing has better performance in response time, scalability and resource utilization. However, the implementation is the most complicated of the three. Therefore, which load balancing algorithm should be used depends on the type of service provided by the system and the system environment, so that the system performance can be effectively improved.

3 A Novel NAT-based Approach for Resource Load Balancing

To solve the loading problem of the TURN server in the ICE architecture under real-time streaming, this method will take the following points as design considerations:

1. Keep the contents of the ICE agreement.
2. Avoid the bottleneck caused by a single network node.
3. Avoid causing additional burden on other servers.
4. Minimize the delay during video streaming.

A controller is added to the traditional ICE architecture to monitor the status of the TURN server. Before the user starts streaming, we calculate the loading value for each TURN server through the controller, and then calculate the loading value based on the load. Value to select the appropriate TURN server, so as to prevent all users from completing load balancing through a single node, and implement the load balancing function on the controller, so that other servers retain the original functions and processes, thereby avoiding Other servers generate additional load. In addition, a fog computing architecture is added to allow users to use the nearest server as a relay to reduce the delay in streaming.

3.1 Approach Aarchitecture

This method divides the overall architecture from top to bottom into Cloud Layer, Fog Layer, and User Layer. As shown in Figure 1, the cloud layer includes STUN servers, signal servers, and TURN server in the cloud; there are multiple layers in the fog layer, and each layer can have multiple nodes. The node structure can be seen in Figure 2. Each node contains a controller and several TURN servers. Each node and level will follow The geographical area is allocated. Taking Figure 1 as an example, node A provides services to users in area A, node B provides services to users in area B, and node C is the upper node of nodes A and B, and provides services to areas A and B. Users in area B, when users in area A want to communicate with users in the same area, they will preferentially relay through the TURN server in node A. The same is true in other areas, but when used in area A When users want to communicate with users in area B, they will first use the TURN server in node C to complete the relay. The purpose of this method is to prevent users connecting across areas from being concentrated on one server. And the server location is as equal as possible to the two ends to reduce the delay inconsistency between the two parties; the user layer includes users who want to communicate with each other. In this architecture, the ICE process is used to achieve NAT traversal, and then to the remote the end user establishes a connection and completes the stream.

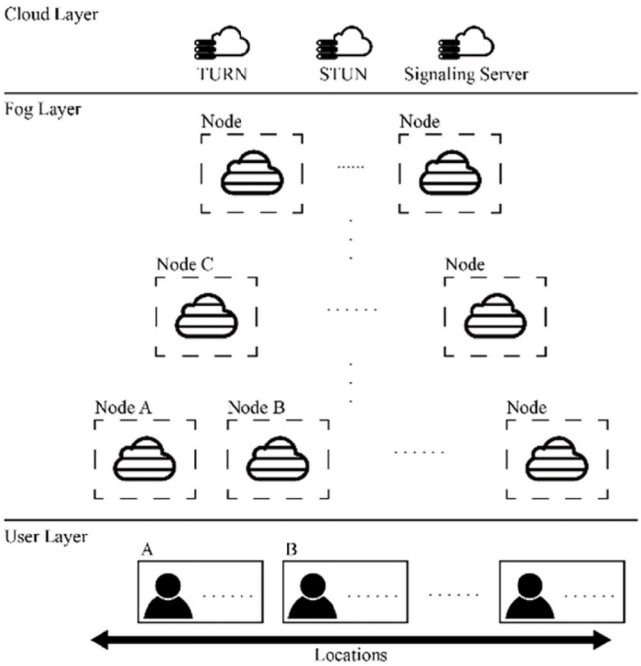


Figure 1. Approach architecture

Figure 2. Node diagram

3.2 Approach Work-flow

3.2.1 End User

The user is at the user level, and its workflow is shown in Figure 3. The user initiates a connection to the remote user in the login state. When the user enters the remote user ID, it will be regarded as initiating communication. Before starting communication, the ICE process will be executed to achieve firewall traversal. First, the user will obtain the external information of his own NAT from STUN, including IP address, Port, and NAT type, and then pass his own NAT information through The signal server exchanges with the remote user and then according to the obtained remote NAT information, it is determined to connect through a peer-to-peer network or initiated a request to the controller, obtain a suitable TURN server by the controller, and use the TURN server as a relay to forward multimedia streams to achieve NAT traversal. Here, if the user is from The IP address received by the controller is the controller address, which means that the TURN that the user can use is at a different node than the controller that originally sent the request. This situation will happen when all TURN servers that are responsible for relaying streaming in this area no longer exist. At this time, the user load will be allocated to the upper-level TURN server. In order to prevent the user from generating unnecessary connections to the underlying controller, when this situation occurs, the user will be disconnected from the previous controller. Then re-establish a connection to the upper-level controller. After re-establishing the connection, the user will re-execute the process of obtaining TURN server information.

In addition, in order to increase the number of user connections to the TURN server and share more of the cloud server load, we have additionally proposed a mechanism to limit the user data transfer rate, the purpose of which is to reduce the number of users being caused by the TURN server Load so that more users can complete the relay through the TURN server in the fog layer. This mechanism will be implemented on the controller when the user requests the TURN server information from the controller.

3.2.2 Signaling Server

The signal server is located in the cloud layer of this architecture. It is responsible for exchanging user information and user authority management. The process is shown in Figure 4. The user must log in first and obtain the authority before requesting to exchange connections with remote users. Line information. The difference from most signal servers is that because this method adds a controller to the architecture, when the user wants to establish a connection through the TURN server, he needs to know what to do Where to obtain the available TURN server information, so the signal server will use the user's IP to identify the area, and then reply the corresponding controller information to the user. In order to make the node maintenance in the framework more flexible, this method stores the controller list in the signal server instead of the client, so that the user can obtain the controller information from the signal server when necessary. To change the organization, you only need to change the controller list in the signal server, which will help maintain and manage the structure in the future.

```
User
INPUT :      local_user_ID, remote_user_ID
1.    loginSignalingServer ( local_user_ID )
2.    requestCommunication ( remote_user_ID )
3.    nat_info = stunProcedure ( )
4.    remote_user_info = exchangeConnectionInfo ( nat_info )
5.    IF ( checkP2PConnection ( remote_user_info ) )
6.      startStream ( remote_user_info )
7.    ELSE
8.      controller_IP = getControllerIP ( )
9.      turn_info = getTURNInfo ( controller_IP )
11.     IF ( isTURNIP ( turn_info ) )
12.       streamByTURN ( remote_user_info, turn_info )
13.     ELSE IF ( isControllerIP ( turn_info ) )
14.       turn_info = getTURNInfo ( turn_info )
15.       streamByTURN ( remote_user_info, turn_info )
```

Figure 3. Workflow of end user

```
Signaling Server
1.    WHILE ( True )
2.      request, user_info = waitingRequest( )
3.      IF ( isLogin ( request ) )
4.        addUser( user_info )
5.      ELSE IF ( isControllerIP ( request ) )
6.        location = checkUserLocation ( user_info )
7.        controller_IP = getControllerIP( location )
8.        respondUser ( controller_IP )
9.      ELSE IF ( isConnectionRequest ( request ) )
10.       sendRequest ( user_info )
```

Figure 4. Workflow of signaling server

3.2.3 Traversal Using Relay NAT

An online notification of the TURN server is used to update the TURN server list of the controller. As shown in Figure 5, when the current load is successfully sent to the controller, it will start to monitor the relay process and set a timer, whenever a group of users relay through the TURN server, the monitoring program will send the load to the controller for update. If there is no user connection for a period of

time, causing the timer to return to zero, the monitoring program will also send the load to the controller to inform the controller of the current TURN servo Whenever the monitoring program sends load information, the monitoring program will reset the timer.

TURN Server Resource Monitor

```
1.    max_bandwidth = getMaxBandwidth ( )
2.    sendServerLoading ( max_bandwidth )
3.    timer = initializeTimer ( )

4.    WHILE ( True )
5.      IF ( newRelay ( ) OR isTimerZero ( timer ) )
6.         sendServerLoading ( max_bandwidth )
7.         resetTimer ( timer )
```

Figure 5. Workflow of TURN server resource monitor

Except for the monitoring program, the TURN server is responsible for the same tasks and processes as the traditional TURN server. Its work and flow can be seen [5]. Because of this, this method can be easily combined with traditional ICE and does not require Redesign the ICE protocol.

3.2.4 Controller for Fog Node

This method has a controller in each node of the fog layer, and each controller provides the following functions:

1. Monitor the resource usage of all TURN servers in the node.
2. Find available TURN servers for users.

And in order to spread the load from the cloud to the fog layer as much as possible, we additionally propose a mechanism to limit the data transfer rate of the user on the TURN server. However, this mechanism will increase the execution time of the controller and cause the establishment of a connection.

The process of monitoring the resource usage of the TURN server is shown in Figure 6. When the TURN server is started, there is a new relay program, or the timer is reset to zero, the load information will be sent to the controller, and the controller will receive After the notification, the TURN server list will be checked. At this time, if the TURN server does not exist in the list, the controller will add the TURN server information and update the list. If the load information returned by the TURN server is received, The TURN server is already in the list, and the controller will update the server information in the list, as shown in equation 1, where P is the set of load values of all TURN servers.

The load value recorded in the table is the memory utilization rate and bandwidth utilization rate returned by the TURN server. As shown in equation 2, where and are respectively the memory usage Rate and bandwidth usage rate, the load value can be obtained

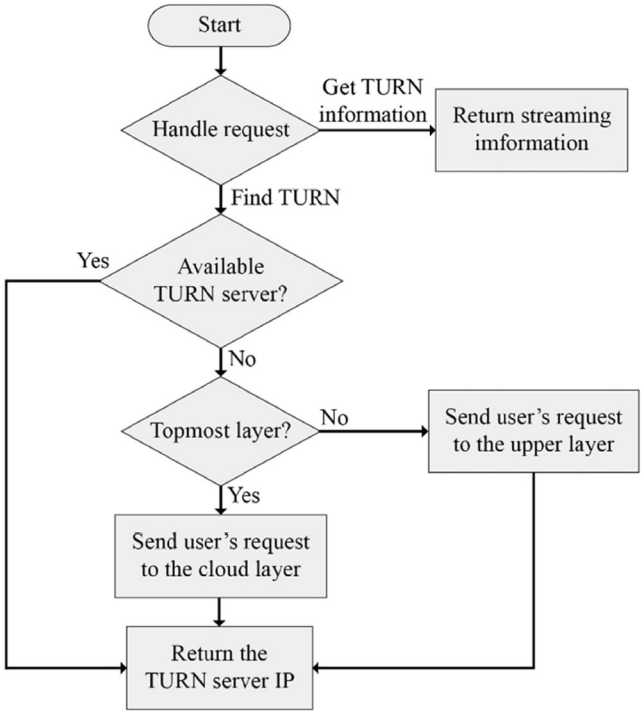


Figure 6. Diagram of fog node controller

after the analysis model calculation, the load value is presented as a probability value, the larger the value, the greater the possibility of the TURN server reaching the load, and vice versa, and the load value will be the choice the basis of TURN server.

$$P = \{p_0, p_1, \ldots, p_i\}, \forall i \in \mathbb{N} \quad \textbf{(1)}$$

$$p_i = A(l_{m/M}, l_{b/B}) \quad \textbf{(2)}$$

$$p_{ai} = p_i < t, 0 < t < 100 \quad \textbf{(3)}$$

$$P_a = \{p_{a0}, p_{a1}, \ldots, p_{0i}\}, \forall i \in \mathbb{N} \quad \textbf{(4)}$$

$$T = \min(Pa) \quad \textbf{(5)}$$

The next page is the process of providing the TURN server available to the user. When the controller receives the user's request, the controller will first check whether there is an available TURN server in the node where it is located, the inspection method is to use the preset threshold as the filter condition to filter the load value obtained after the monitoring process is analyzed in the TURN server information. If the load value exceeds the threshold, the TURN server will be regarded as the maximum Load, but if the load value does not exceed the threshold, it will be compared with other load values that do not exceed the threshold, and the TURN server with the smallest load value will be used to return the user. For the load value of the TURN server that exceeds the threshold t, equation 4 is the set of all load values that do not exceed the threshold. After comparison, the smallest value will be selected as the basis for returning the TURN server IP, as shown

in equation 5. But if all the load values in the node have exceeded the threshold, the controller will first determine whether the node is at the top of the fog layer, and if it is the top node, it will send the user request up to the cloud server to complete the relay If the node is not at the top level, the user request will be sent to the upper controller, and after waiting for the upper controller to reply the controller IP address, the IP address will be returned to the user, allowing the user to directly create the target controller Connect to obtain the address of the usable TURN server and the stream information on the TURN server, so that the user can use the TURN server to complete the relay and communicate with remote users.

In order to have a better load balancing effect, so that the TURN server in the node can carry more users, we propose to limit data transmission to the users of the node where the controller is located before sending the user request to the upper layer. When all the load values in the node have exceeded the threshold, the TURN server with the highest transmission rate will be selected as the choice to limit the TURN The data transmission rate of the user on the server to release server resources. If the data transmission rate of all TURN servers has been limited to the lowest point set, it will first check whether the server is Located in the uppermost layer of the fog layer, if it is the uppermost node, the user request will be sent to the cloud to complete the relay. If it is not the uppermost node in the fog layer, the user request will be sent to the controller of the upper node, so that The upper-level node selects the available TURN server according to the process, and waits for the upper-level controller to reply to the controller IP address, and then returns the IP address to the user, allowing the user to directly establish a connection to the target controller and obtain the available ones The address of the TURN server and the streaming information in the TURN server will finally allow the user to complete the relay through the TURN server and communicate with remote users.

4 Experiment Results

The experiments in this study can be divided into three parts, discussing the resource loading, the analysis when the controller selects the TURN server, and the delay time added by the controller. The experiment content is as follows:

1. Obtain the maximum load of the TURN server in the experimental environment.
2. Compare the accuracy of different analysis methods.

Compare the delay time added by the controller in the process with or without limiting the data transfer rate.

The experimental environment can be seen in Figure 7. We set the environment in the fog layer as single-layer, double-layer, three-layer, four-layer, and five-layer architecture. Among them, node 1 is located in the first layer, and node 2 is located in the second layer. By analogy, there is one controller in each node, and the information of five TURN servers in each controller is stored in the controller list. The signal server and STUN server use server 1, and the controller in the node depends on which The node numbers use server 2, server 3, server 4, server 5, and server 6. In addition, users will connect through NAT, and the NAT type is set to symmetric NAT.

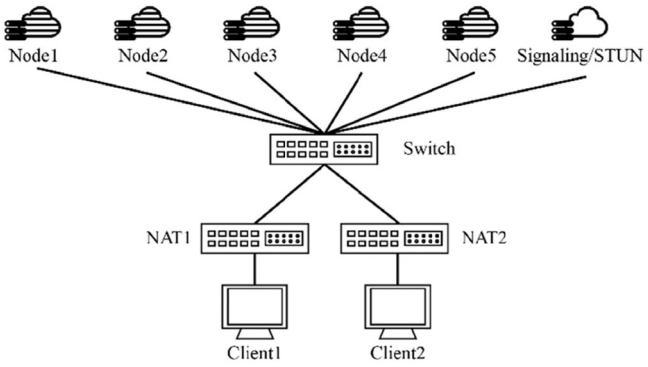


Figure 7. Experimental environment

4.1 Maximum Loading

We increase the load on the TURN server according to the architecture. The relationship between the memory and data transfer rate of each resolution can be seen in Figure 8. In this experimental environment, the maximum server bandwidth is about 94.15 Mbps. When you reach the highest point, although you can continue to add users, there will be a delay in establishing a connection. Therefore, subsequent experiments will regard the maximum bandwidth of the server as the upper limit of the server network resource usage and the maximum memory usage About 99.17%, but when the memory usage reaches about 95%, some trips will be ended to release resources. In order not to affect the original running schedule on the server, we set the maximum usage rate before the system releases resources Treated as the upper limit of memory resources.

4.2 Controller Latency

It can be seen from Figure 9 that this method uses the controller to select the TURN server. In the experimental environment, the average response time of a single-layer node is about 0.088 seconds, and an average increase of about 0.128 seconds for each additional layer of nodes, to the fifth layer of nodes. The average time required to obtain the TURN server address increased to about 0.602 seconds. In addition, we have listed the maximum and minimum time required to obtain the TURN server information for each layer.

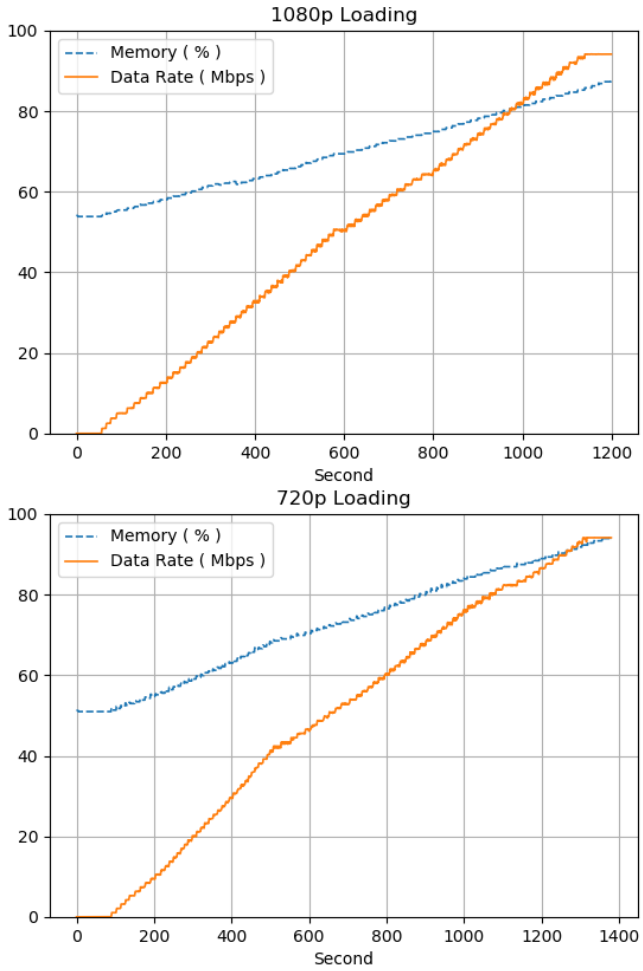


Figure 8. Maximum load

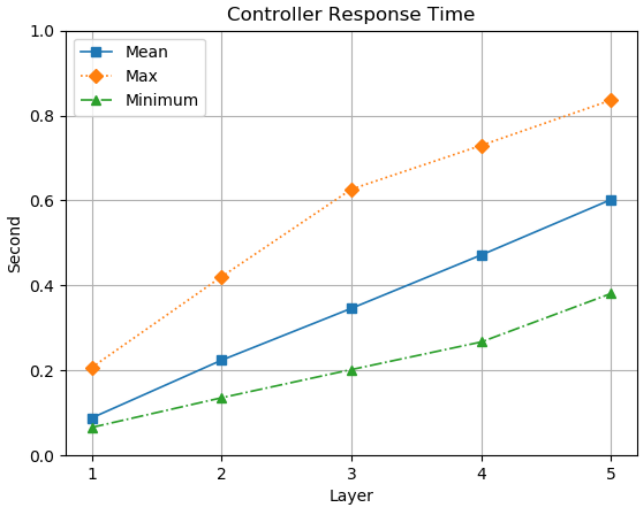


Figure 9. Controller latency

5 Conclusion

In the load balancing method proposed in this paper, we use ICE as the basis, use the fog computing architecture to distribute TURN servers to the edge of the network, and add controllers to the fog layer nodes to monitor resources through the TURN server. The program manages the resources of the TURN server. When the user cannot establish a peer-to-peer connection through the STUN server, it sends a request to the controller to obtain the available TURN server information. Here we use the analysis model to obtain the TURN server information Load value. The controller uses the load value to determine which TURN server information should be responded to.

In the experiment, we provide the maximum load that the hardware environment can withstand for reference, and test the delay added by the controller in the method, and know that the delay caused by the controller itself is very small, and when choosing TURN servo The addition of a mechanism to limit the user data transfer rate in the server process does not add too much delay time, and can allow the fog layer TURN server to spread more cloud load. Summarizing the above research results, the method proposed in this paper can not only distribute the load of the cloud, but also does not transfer the server load to the user, which will cause an additional burden on the user environment.

References

[1] J. Rosenberg, *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*, IETF, RFC5245, April, 2010.

[2] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, *Session Traversal Utilities for NAT (STUN)*, IETF, RFC5389, October, 2008.

[3] R. Mahy, P. Matthews, J. Rosenberg, *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*, IETF, RFC5766, April, 2010.

[4] U. Sukhee, *The Research and Implementation of NAT Traversal for RTSP*, Electrical Engineering and Computer Science, National Taipei University of Technology, 2012.

[5] K. I. Z. Apu, N. Mahmud, F. Hasan, S. H. Sagar, P2P Video Conferencing System Based on WebRTC, in *IEEE International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Cox's Bazar, Bangladesh, 2017, pp. 557-561.

[6] Y. Wei, D. Yamada, S. Yoshida, S. Goto, A new method for Symmetric NAT Traversal in UDP and TCP, in *Asia Pacific Advanced Network 2008*, Queenstown, New Zealand, 2008, pp. 1-8.

[7] K. Tam, H. Goh, Session initiation protocol, in *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT'02.*, Bankok, Thailand, 2002, pp. 1310-1314.

[8] WebRTC, 2018, https://webrtc.org/.

[9] D. Zhang, C. Zheng, H. Zhang, H. Yu, Identification and Analysis of Skype Peer-to-peer Traffic, in *2010 Fifth International Conference on Internet and Web Applications and Services (ICIW)*, Barcelona, Spain, 2010, pp. 200-206.

[10] W.-E. Chen, Y.-B. Lin, A.-C. Pang, An IPv4-IPv6 Translation mechanism for SIP Overlay Network in UMTS

All-IP Environment, *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 11, pp. 2152-2160, November, 2005.

[11] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, M. Bhatia, *Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments*, IETF, RFC5853, April, 2010.

[12] B. A. Miller, T. Nixon, C. Tai, M. D. Wood, Home Networking with Universal Plug and Play, *IEEE Communications Magazine*, Vol. 39, No. 12, pp. 104-109, December, 2001.

[13] F. Shang, L. Mao, W. Gong, Service-aware Adaptive Link Load balancing Mechanism for Software-Defined Networking, *Future Generation Computer Systems*, Vol. 81, pp. 452-464, April, 2018.

[14] M. M. Rathore, H. Son, A. Ahmad, A. Paul, G. Jeon, Real-time Big Data Stream Processing Using GPU with Spark Over Hadoop Ecosystem, *International Journal of Parallel Programming*, Vol. 46, No. 3, pp. 630-646, June, 2018.

[15] J. Wang, M. Qiu, B. Guo, Enabling Real-time Information Service on Telehealth System over Cloud-based Big Data Platform, *Journal of Systems Architecture*, Vol. 72, pp. 69-79, January, 2017.

[16] B. Shu, H. Chen, M. Sun, Dynamic Load Balancing and Channel Strategy for Apache Flume Collecting Real-Time Data Stream, in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, Guangzhou, China, 2017, pp. 542-549.

[17] Y. Ohshima, B. Freudenberg, D. Amelang, Kanto: A Multi-participant Screen-sharing System for Etoys, Snap!, and GP, in *Proceedings of the 3rd ACM SIGPLAN International Workshop on Programming Experience*, Vancouver, BC, Canada, 2017, pp. 7-10.

[18] I. Masuda-Katsuse, Remote Articulation Test System based on WebRTC, in *Proc. Interspeech 2017*, Stockholm, Sweden, 2017, pp. 4030-4031.

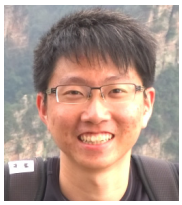
[19] P. Saint-Andre, *Jingle ICE Transport Method*, XMPP.org, 2017.

[20] H. She, O. Wittenberg, I. Warren, An Ad Hoc Broadcasting Application by Way of Mobile Devices, in *Proceedings of the Australasian Computer Science Week Multiconference*, Geelong, Australia, 2017, Article No. 21.

Biographies

Chin-Feng Lai is a professor at Department of Engineering Science, National Cheng Kung University and Department of Computer Science and Information Engineering, National Chung Cheng University since 2016. He received the Ph.D. degree in Department of Engineering Science from National Cheng Kung University, Taiwan, in 2008. He received Best Paper Award from IEEE 17th CCSE, 2014 International Conference on Cloud Computing, IEEE 10th EUC, IEEE 12th CIT. He has more than 100 paper publications and 9 papers selected to TOP 1% most cited articles by Essential Science Indicators (ESI). He is an associate editor-in-chief for Journal of Internet Technology and serves as editor or associate editor for IET Networks, International Journal of Internet Protocol Technology, KSII Transactions on Internet, Information Systems and Journal of Internet Technology. His research focuses on Internet of Things, Big Data Analysis and Edge Computing etc. He is an IEEE Senior Member since 2014.

Hung-Yen Weng received the master degree in Engineering Science from National Cheng-Kung University in 2009. He had been working in HTC for 7 years since 2010 and then joined Google in 2018. He works mainly on system software of baseband OS on mobile phones. He has contributed to many flagship smart phones, such as HTC M7/M8/M9/10/U11 and Google Pixel/Pixel2. He is a PhD student in Engineering Science from National Cheng-Kung University.



Hao-Yu Chou received the master degree in Department of Computer Science & Information Engineering from National Chung Cheng University in 2018. He has been working in Sercomm Corporation, Taiwan for broadband product developments.

Yueh-Min Huang is a Chair Professor in Department of Engineering Science, National Cheng-Kung University, Taiwan. His research interests include e-Learning, multimedia communications, and artificial intelligence. He received his MS and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1988 and 1991 respectively. He has co-authored 3 books and has published more than 280 refereed journal research papers. Dr. Huang has received many research awards, such as Taiwan's National Outstanding Research Award in 2011/2014, as well as 2017 Taiwan Outstanding IT Elite Award. He has completed over 60 Ph.D. and 300 MS thesis students. Dr. Huang is in the editorial board of several international journals in the area of educational technology, computer communications, and web intelligence. Dr. Huang is also the funding chair of International Symposium of Emerging Technologies for Education (SETE) and International Conference of Innovative Technologies and Learning (ICITL). Dr. Huang is a senior member of the IEEE and became Fellow of British Computer Society in 2011.