SDN-based Implementation of P2P Streaming Networks with Dynamic Reconfiguration

Ryo Shibasaki, Noriko Matsumoto, Norihiko Yoshida
Graduate School of Science and Engineering
Saitama University
Saitama, Japan
Emails: {shibasaki, noriko, yoshida}@ss.ics.saitama-u.ac.jp

***Abstract*—The Software Defined Networking (SDN) technology is one of the major technical infrastructures for the digital society. This paper focuses on a dynamic streaming network and its efficient implementation. A Peer to Peer (P2P) streaming network tends to be unbalanced because of the dynamic nature of its nodes to join and leave, and often suffers from delivery delay. Therefore, it is required to reconfigure the network dynamically to keep it balanced. This paper proposes an SDN-based scheme for adaptive reconfiguration of the structure and routing in a P2P streaming network. This scheme enables fully decentralized delivery under a centralized control. Some simulation-based experiments confirmed that this scheme worked effectively.**

Keywords*—*P2P streaming; SDN

I. INTRODUCTION

Streaming is used to download a multimedia content while playing it. In this delivery method, the content is divided into units called segments. Segments are transferred from the server to the clients in turn. Typically, the Client/Server (C/S) model is available for streaming. However, the C/S model has problems such as the increase of the server load and maintenance costs according to the increase of the number of clients. Peer-to-Peer (P2P) Streaming has attracted attention in order to solve these problems. However, P2P streaming also has shortcomings such as failures due to concentration of the connection to a specific node and delays in receiving contents due to the increase of the number of hops from the source node. In order to prevent these shortcomings, it is important to reconfigure the P2P streaming network dynamically to keep it balanced [1]. Traditionally, each node in a P2P network performs data transfer and routing in an autonomous distributed manner. Load balancing could be achieved by each node independently, however it would be inefficient. Therefore, we propose a centralized control scheme for such decentralized P2P networks.

Recently, Software Defined Networking (SDN) is attracted attention as a concept of centralized control of network devices. SDN can control network devices flexibly using software. SDN has already been used in Google's data center [2] for example. In addition to this, SDN is expected to realize new technologies such as Network Function Virtualization (NFV) and Information Centric Network (ICN) [3]. OpenFlow [4] is a typical implementation technology of SDN. The OpenFlow specifies network devices from the datalink layer to the transport layer. However, a conventional P2P network is implemented as an overlay network at the application layer.

In this paper, we propose a scheme to configure a P2P streaming network at the IP layer using OpenFlow. In our proposal, the OpenFlow configures a P2P streaming network and a centralized routing control for network devices. The structure of this paper is as follows: Section II introduces a routing method of P2P network using OpenFlow. Section III describes our proposal. Section IV shows some results of simulation-based experiments and related considerations. Finally, Section V includes some concluding remarks and future work.

II. RELATED WORKS

OpenFlow consists of a controller, OpenFlow switches and OpenFlow Protocol. The OpenFlow switches forward or discard packets according to "flows" which are given from the controller. The OpenFlow Protocol is used to communicate between the controller and switches. The controller is implemented in software, and can update flows for the switches dynamically. The network topology and its routing mechanism is reconfigured flexibly and dynamically under the centralized control in this manner. The routing method for the P2P network using OpenFlow will be presented later.

In streaming delivery, delay or loss in the reception of the segments leads to interruption of playback of contents. Trajikovska et al. [5] proposed SDN based Quality-of-Service (QoS) control in a P2P streaming network. In this proposal, the provider decides a parent node for a new node which joins the P2P streaming network. First, the provider figures out candidate nodes which are close to the new node using the Round-Trip-Time (RTT) distance between the new one and others. Next, the provider checks the bandwidth between the new one and the parent candidates. The new node is connected to the parent node which is selected on the basis of the RTT distance and the bandwidth. After that, the bandwidth is adjusted between these nodes using OpenFlow meter table.

Othman et al. [6] proposed a method that the controller executes routing control. In this proposal, changing from the C/S model to the P2P model occurs depending on the server load. When changing to the P2P model, the server sends a request to redirect content requests to the controller. The redirection request includes the content IDs and the list of nodes which receive this content. This list contains the IP address of the node and the number of child nodes which can be connected to this node. When the controller receives the redirection request, the controller sends some necessary flows

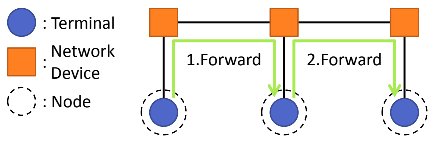


Figure 1. Content delivery of a traditional P2P network

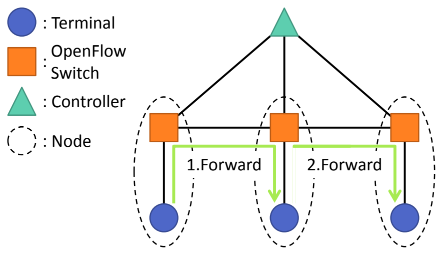


Figure 2. Content delivery of a P2P network using OpenFlow

to the OpenFlow switches. A content request is then redirected to a node which already has the content.

These proposals focus only on the situation when a node joins a P2P network. Additionally, some studies proposed reconfiguration of a P2P network dynamically. Akiyama et al. [7] applied the publish and subscribe (pub/sub) model to a P2P network. In the pub/sub model, a publisher sends a content to the middleware, and the content is categorized in some topic. Then the middleware sends the content to subscribers which are joined to the topic. In this proposal, the controller examines the physical topology using Link Layer Discovery Protocol (LLDP). The agent which runs on the node is migrated based on the physical topology and the topic name.

III. Proposed Method

This section introduces our proposal. Our proposal is based on the study by Ono et al. [1]. Ono et al. proposed a scheme to reconfigure a P2P streaming network dynamically. To implement this, each node monitors its neighbor nodes such as its parent and children. If the parent node can spare its capacity, a child node reconnects itself to the parent. Load balancing is achieved in this manner, and delay in content delivery is reduced. In this proposal, each node in a P2P streaming network performs data transfer and routing in an autonomous distributed manner. However, this kind of autonomous distributed manner is inefficient because each node performs a similar behavior and causes excessive control packets.

There are two kinds of streaming: on-demand and live delivery. In on-demand streaming, a content created in advance is delivered to clients when its server receives a content request. In live streaming, a content is delivered to clients in real time while recording and encoding. In traditional P2P live streaming, each leaf node which is joined to the P2P streaming network receives and sents the content so that all the nodes can decode and play the same content at the same time (Figure 1).

Since the same content is delivered at the same time, P2P live streaming has some resemblance with IP multicast by network devices (Figure 2). In this paper, a pair of a host and its corresponding switch is considered a "node."

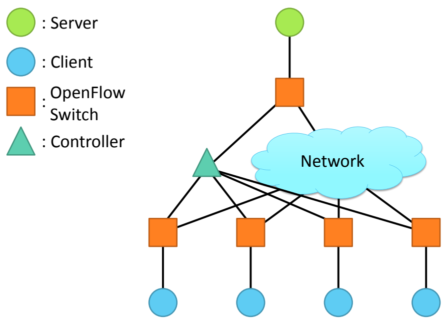


Figure 3. Proposal system overview

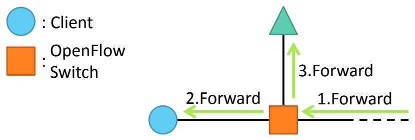
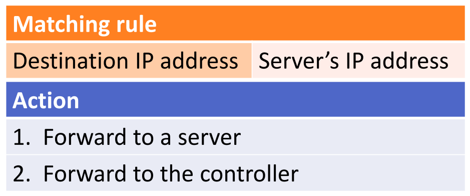


Figure 4. A content request is forwarded to the controller



Matching rule	
Destination IP address	Server's IP address
Action	
1. Forward to a server	
2. Forward to the controller	

Figure 5. The flow to forward a content request to the controller

In our proposal, the controller manages the route of the P2P streaming network. Our proposal comprises a server, clients, OpenFlow switches and the controller (Figure 3). The server forwards segments to clients according to a content request. The clients receive segments from a server. If a client has child nodes, the client forwards its segments to the children. The OpenFlow switch forwards a content request and segments according to the flows which are given from the controller. The controller sends flows so as to change the P2P streaming network. The procedure of the controller is described below.

A. *Management of nodes*

In our proposal, the controller manages nodes in the P2P streaming network in a centralized manner. To know a node which has joined or left from the P2P streaming network, a content request is forwarded to the controller (Figure 4). To forward to the controller, the flow which is shown in Figure 5 is given to an OpenFlow switch neighboring to the server. When the controller receives a content request, the controller reads the header of its packet to get the source IP address. After that, the controller updates the network topology information within it according to join or leave of the node of the IP address.

B. *Delivery of contents*

Segments which are forwarded from a parent node are copied in an OpenFlow switch. The segments are passed to the coupled host as well as transferred to its child nodes (Figure 6). To implement this, the flow which is shown in Figure 7 is kept in the OpenFlow switch. According to this flow, the client assumes that segments are forwarded from a server in spite that

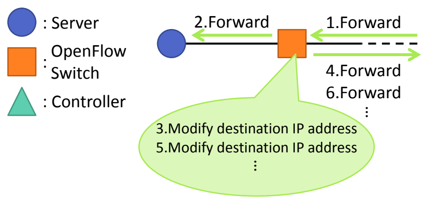
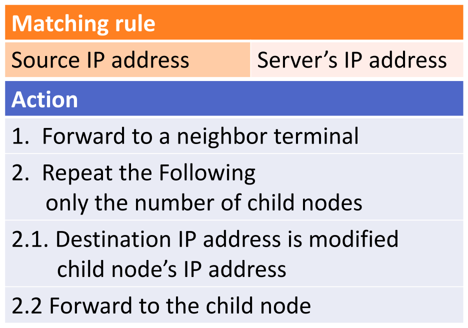


Figure 6. Delivery of segments



Matching rule	
Source IP address	Server's IP address
Action	
1. Forward to a neighbor terminal	
2. Repeat the Following only the number of child nodes	
2.1. Destination IP address is modified child node's IP address	
2.2 Forward to the child node	

Figure 7. The flow to forward a content to child nodes

actually they are forwarded from the parent node. This delivery is similar to IP multicast using OpenFlow [8][9], however IP addresses are not necessary unlike IP multicast. Accordingly, the packet which is copied by this flow is forwarded on a network similar to ordinary packets.

C. Reconstruction of a P2P stereaming network

The controller must select a parent node of a new node which has joined the P2P streaming network. Similarly, the controller must select a parent node of nodes which were child nodes of a leaving node. A parent node is determined following the below procedure.

1) Select candidate nodes which is located at the lowest depth from the server among nodes which can spare their capasities.
2) Select a parent node which has the smallest number of child nodes out of the candidates.

Following these steps, a joining node or child nodes of a leaving node select a parent node to reconstruct a balanced P2P streaming network. However, the P2P streaming network may be unbalanced because this procedure does not consider descendant nodes of these nodes. Therefore, the controller checks the P2P streaming network topology regularly. If the P2P streaming network is unbalanced, an unbalanced node selects a parent node again using the above procedure. After that, the P2P streaming network is reconfigured to be balanced it. An unbalanced node is determined using (1).

In our proposal, the maximum number of child nodes which can be connected to a node is fixed to a constant (c). The total number of clients (n_{max}) which can be connected in the depth d is shown below when the server's depth is 0.

$$\begin{aligned} n_{max} &= \sum_{i=1}^{d} c^i \\ &= \frac{c(c^d-1)}{c-1} \end{aligned}$$

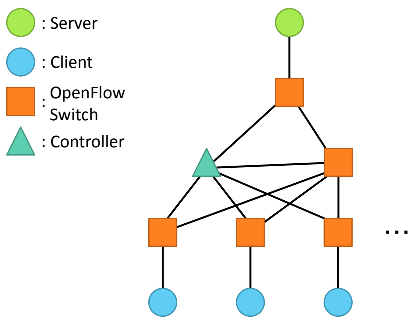


Figure 8. A virtual network used in the experiment

TABLE I. The parameters used in the experiment

Parameter	Value
The number of child nodes which are connected the node	2
The number of segments to be sent	10 packets per second
Check interval for the network topology	every 2 seconds
Interval for statistics of OpenFlow switches	every 1 second

Therefore, if the number of joining clients to the P2P streaming network is n, the optimal value of the depth of the P2P network is shown below.

$$\begin{aligned} n &\leq n_{max} \\ \frac{\ln(nc-n+c)}{\ln c} - 1 &\leq d \end{aligned} \qquad (1)$$

The optimal value is the minimum value of d which satisfies (1). If all the unbalanced nodes select a parent node again at once, the P2P streaming network changes drastically. As a result, the loss or the duplication of segments may occur. Furthermore, the reconfiguration does not complete by the time of the next check because processing time of the controller is increased. From the above, the depth of nodes which select a new parent node are restricted to $d_{min}+1$.

The controller must send flows, which are shown in Figure 7, to the OpenFlow switches when the P2P streaming network topology is changed. In our proposal, segments that are forwarded from a parent node is considered equivalent to the ones forwarded from a server. Accordingly, if a parent node which is connected to the node is changed, the node does not have any influence. However, if child nodes which are connected to the node are changed, the node must send segments to these new nodes. The controller must send the flow to these nodes' switch to forward segments to child nodes. From the above, the OpenFlow switches which must update the flow are the old parent node and the new parent node.

IV. Evaluation

A simulator was made for experiments in order to confirm the effectiveness of our proposal.

A. Simulator Design

The simulator was implemented in Trema which is one of the OpenFlow frameworks. The experiment was performed using a virtual network on Trema. A virtual network is composed of star topology (Figure 8). The controller detects links between OpenFlow switches using LLDP in order to forward packets. Links between an OpenFlow switch and a leaf are detected using Packet-In messages. Packets are forwarded using Dijkstra's algorithm. The parameters used in

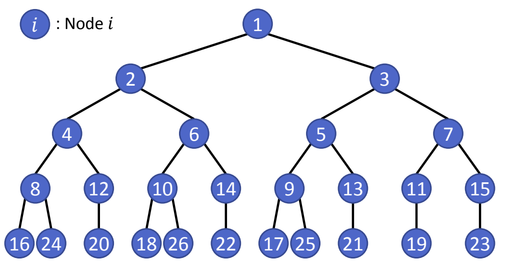


Figure 9. The P2P streaming network topology before nodes join

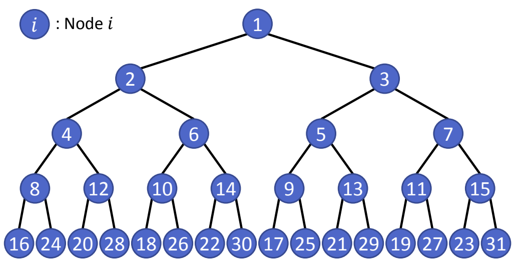


Figure 10. The balanced P2P streaming network topology

the experiment are shown in Table I. The controller knows a set of Server's IP address and an OpenFlow ID which neighbors the server in advance. A server forwards segments to the OpenFlow switch. The OpenFlow switch forwards segments according to the flow shown Figure 7. The experiment was performed as described below under these conditions.

- Join of nodes
  We examined the P2P streaming network topology when nodes join the network. Furthermore, we examined whether each client receives all the segments or not.
- Withdrawal of a node
  We examined the P2P streaming network topology when nodes left the P2P streaming network. Then, we examined the number of nodes which changed its parent node and flows which are forwarded to an OpenFlow switch from the controller. Furthermore, we examined whether each client receives all the segments or not.
- Comparison of the P2P model and the C/S model
  If the number of child nodes which are connected to a node is unlimited, our proposal constructs a network similar to the C/S model. Accordingly, we compared with the C/S model along with the P2P model and examined each client whether to receive segments or not. And then, we compared the load of OpenFlow switches in the P2P model and the C/S model.

The amount of segments which are sent from a server and received by the clients was compared in order to confirm examine that each client receives all segments or not. The amount of segments which are sent or received is obtained using statistics per port of an OpenFlow switch. However, the amount of segments which are sent from the server and received by the clients are not equal when the node joins or leaves in the P2P streaming network. Accordingly, we calculated the amount of segments per second and compared their values. The load of an OpenFlow switch is the sum of the number of packets which are sent from its switch and received by the switch.

B. Results and Considerations

- Joining of nodes

A node joined in the P2P streaming network which already includes a server and 25 clients (Figure 9) every 10 seconds. 5 new nodes joined the network in total. As a result, a P2P streaming network is reconfigured to be balanced (Figure 10).

Figure 11 compares the amount of segments which are sent from the server and an average of the amount of segments

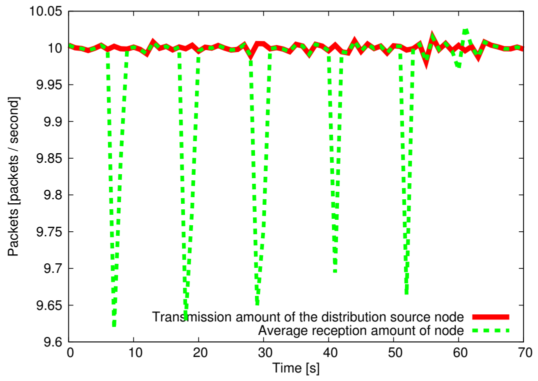


Figure 11. Comparison of the amount of segments which are sent from the server and received by the clients

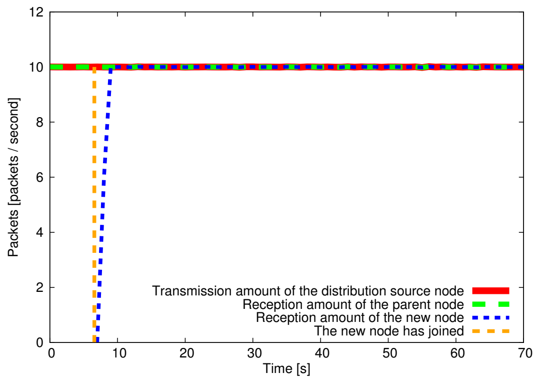


Figure 12. Comparison of the amount of segments which are sent from the server and received by the new node and its parent node

which are received by the clients when 5 nodes join the P2P streaming network. These values are equal in general. However, it is not equal occasionally due to delay such as to modify the flow and forward segments before a joining node begins to receive segments (Figure 12). The amount of segments which are received by its parent node is equal to the amount of segments which are sent from the server. Accordingly, a joining node causes the delay before receiving segments when it joins the P2P streaming network. However, other nodes do not suffer from this.

- Leaving of a node

A node whose depth from a server is 1 is left from the network which has the server and 30 clients (Figure 10). The unbalanced P2P streaming network topology is changed immediately after a node left (Figure 13). Its topology is reconfigured to be balanced by repeating the reconfiguration

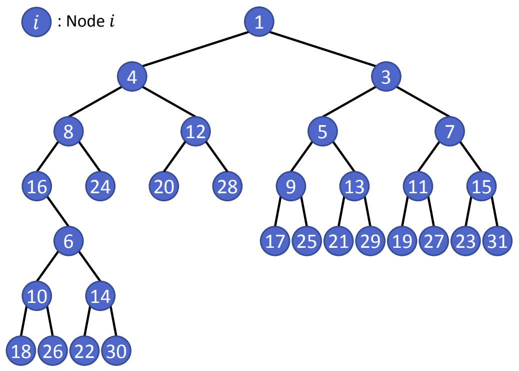


Figure 13. The P2P streaming network which is roconfigured immediately after a node left.

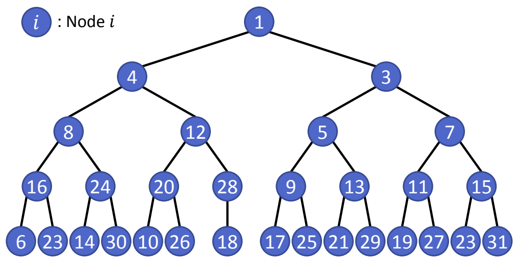


Figure 14. The balanced P2P streaming network after reconfiguration

TABLE II. Difference of the process according to the depth from a server

The depth from a server	1	2	3	4
The number of reconstruction	3	2	1	1
The number of nodes which reselect a parent node	8	4	2	0
The number of flows which are modified	12	6	3	1

procedure (Figure 14). Similar experiments were conducted repeatedly where a node in a different depth left. Then we examined the number of reconfiguration to make the P2P streaming network balanced, the number of nodes which reselect parent nodes and modified flows in the OpenFlow switches (Table II). If a node whose depth is d_n left, these values are as follows. $d = d_{min} - d_n$, where d_{min} is calculated using (1). The number of reconfiguration is d. The number of nodes which reselect parent nodes is c^d, where c is the number of child nodes which are connected to the nodes. Then, the number of flows which are modified in an OpenFlow switch is $c^d + c^{d+1}$.

Figure 15 compares the amount of segments which are sent from the server and an average of the amount of segments which are received by the clients when a node left from the P2P streaming network. These values are equal. Accordingly, other nodes do not suffer from any influence by a leaving node and reconfiguration of the P2P streaming network.

- Comparison of the P2P model and the C/S model

We performed similar experiment which the node joins or leaves to change the number of child nodes is unlimited. Figure 16 compares the amount of segments which are sent from the server and an average of the amount of segments which are received by the clients when 5 nodes join the P2P streaming network. If these values are not equal, a joining node causes delay as mentioned above. Figure 17 compares the amount of segments which are sent from the server and an average of the amount of segments which are received by the clients when a node leaves from the P2P streaming network. If the amount of the segments which are received by the clients is more than the amount of segments which are sent from the server, the leaving node receives segments after it left from the P2P streaming network. From the above, our proposal system would be able to perform also as the C/S model.

Figure 18 compares the P2P model and the C/S model when 5 nodes join. Similarly, Figure 19 compares the P2P model and the C/S model when a node leave. In the C/S model, the load of the OpneFlow switch which is next to the server is increased or decreased according to the number of clients.

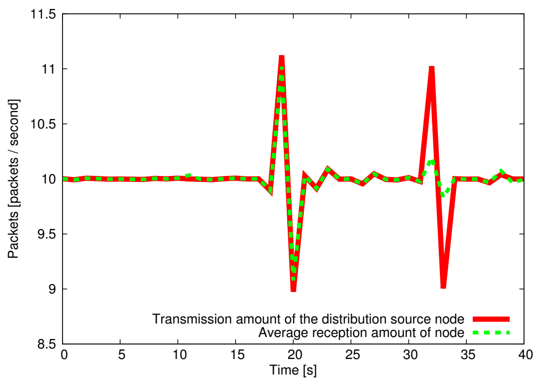


Figure 15. Comparison of the amount of segments which are sent from the server and received by the clients

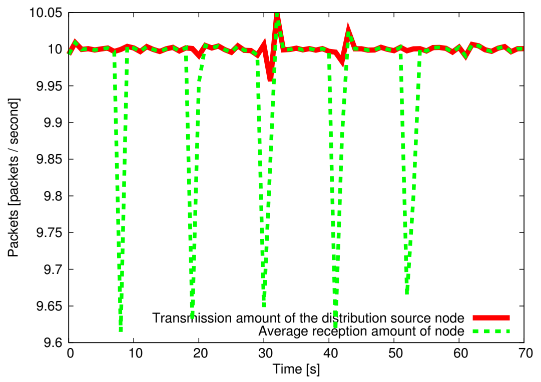


Figure 16. Comparison of the amount of segments which are sent from the server and received by the clients when 5 nodes join

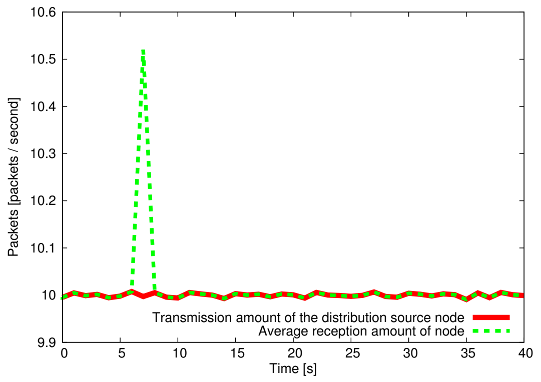


Figure 17. Comparisone of the amount of segments which are sent from the server and received by the clients when a node leaves

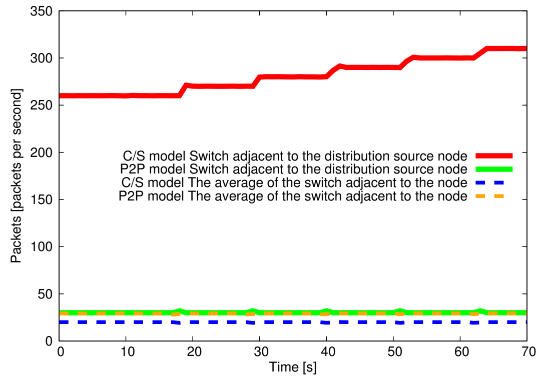


Figure 18. Comparison of the load of swithes when 5 nodes join

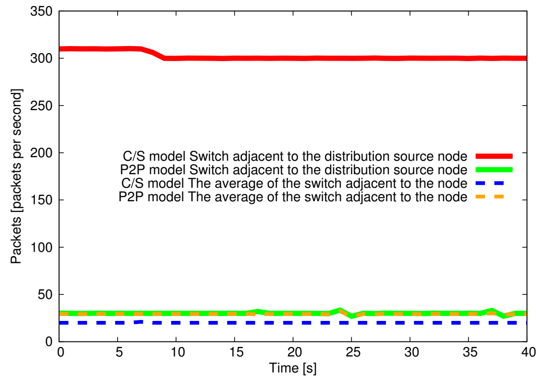


Figure 19. Comparison of the load of swithes when a node leave

On the other hand, in P2P model, the load of the OpneFlow switch which is next to the server is constant. Moreover, we can claim that the P2P model realized load balancing because the loads of all the OpenFlow switches are equal.

V. Conclusion and Future Work

In this paper, we proposed a method which the controller controls routing of the P2P streaming network. In our proposal, the controller reconfigures the P2P streaming network topology when a node joins or leaves. The proposal uses OpenFlow, and can reconfigure more flexibly and dynamically than the implementation which does not use OpenFlow. We conducted experiments using Trema in order to confirm the effectiveness of our proposal. As a result, a node causes delay before receiving segments. However, the node receives all segments.

The disadvantages of our proposal are as below. First, the wrong P2P streaming network topology may be configured due to complex control packets. Next, if OpenFlow switches and non-OpenFlow switches are mixed, the node which has a non-OpenFlow switch should be a leaf node. We will solve these disadvantages.

At last, we describe future works.

- In this paper, we performed experiments using test packets of Trema instead of segments. Therefore, we need to confirm the behavior of the server, clients and OpenFlow switches when segments are forwarded using streaming protocol, such as HTTP Live Streaming [11].
- In OpenFlow, network devices are controlled by the controller in a centralized manner. However, it is difficult to apply OpenFlow to a large network such as the Internet currently. Accordingly, if OpenFlow is applied to the P2P streaming network, the following methods may be needed. First, a virtual switch would be prepared on each host which joins the P2P streaming network. Next, an overlay network is configured using these switches. Lastly, the controller controls these virtual switches.

References

[1] K. Ono, A. Zhygmanovskyi, N. Matsumoto, and N. Yoshida, “Resilient Live-Streaming with Dynamic Reconfiguration of P2P Networks,” EMERGING 2014, The Sixth International Conference on Emerging Network Intelligence, 2014, pp. 6–11.

[2] S. Jain, et al., “B4: Experience with a globally-deployed software defined WAN,” ACM SIGCOMM Comp. Comm. Review. Vol. 43. No. 4. ACM, 2013, pp. 3–14.

[3] B. A. A Nunes, M. Mendonca, X. Nguyenm, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” Comm. Surveys & Tutorials, IEEE, 16.3, 2014, pp. 1617–1634.

[4] N. McKeown et al.,“OpenFlow: Enabling Innovation in Campus Networks,” ACM SIGCOMM Comp. Comm. Review 38.2 pp. 69-74, 2008.

[5] I. Trajikovska, P. Aeschimann, C. Marti, T. M. Bohnert, and J. Salvachtia. “SDN enabled QoS provision for online streaming services in residential ISP networks,” Proc. Consumer Electronics-Taiwan IEEE Int. Conf., 2014, pp. 33–34.

[6] O. M. M. Othman and K. Okamura, “On Demand Content Anycasting to Enhance Content Server Using P2P Network,” IEICE TRANS. on Info. and Sys., 2012, pp. 514–522.

[7] T. Akiyama, K. Iida, J. Zhang, and Y. Shiraishi, “Proposal for a New Generation SDN-Aware Pub/Sub Environment,” Proc. 13th Int. Conf. on Net., 2014, pp. 210–214.

[8] T. Akiyama, Y. Kawai, Y. Teranishi, R. Bannno, and K. Iida, “SAPS: Software Defined Network Aware Pub/Sub – A Design of the Hybrid Architecture Utilizing Distributed and Centralized Multicast,” Proc. Comp. Software and App. Conf. on IEEE 39th Annual, 2015, pp. 361–366.

[9] Y. Nakagawa, K. Hyoudou, and T. Shimizu, “A management method of IP multicast in overlay networks using openflow.” Proc. 1st workshop on Hot topics in software defined networks, ACM, 2012, pp. 91–96.

[10] Trema, http://trema.github.io/trema/ [retrieved: March, 2016]

[11] HTTP Live Streaming, https://developer.apple.com/streaming/ [retrieved: March, 2016]