Study of a method of supporting IP routing for MPTCP by SDN

*

Koki Izumi
Graduate School of Engineering
Nagoya Institute of Technology
Nagoya, Japan
izumi@en.nitech.ac.jp

Yoshihiro Ito
Graduate School of Engineering
Nagoya Institute of Technology
Nagoya, Japan
yoshi@nitech.ac.jp

***Abstract*—This paper proposed a method of improving QoS for MPTCP, which is one of the next generation transport layer protocols. Although MPTCP can realize high speed and stable communication by using multiple paths simultaneously. Paths selected by IP are not always appropriate to MPTCP. Thus, this method supports IP routing for MPTCP using SDN. The authors implement the method and evaluate its effectiveness by experiment.**

I. Introduction

In these days, with the diversification of communication environments, one terminal has been able to use plural access networks simultaneously. For example, many devices such as home appliances and tablet PCs often have more than one network interfaces, such as 4G(LTE) and Wi-Fi.

In order to utilize plural access networks efficiently, MPTCP(MultiPath TCP)[1] has been standardized as a next generation transport layer protocol. MPTCP can realize high speed and stable communication by simultaneously using multiple paths via plural network interfaces.

Here we should discuss paths used by MPTCP. Each MPTCP's path can only select one of existing paths because MPTCP is a transport layer protocol. A network layer protocol must play a role in deciding a end-to-end path; thus IP(Internet Protocol)[2] is responsible for routing. In other words, the path selection of MPTCP and IP routing operate independently to each other. Therefore, when MPTCP selects some paths, MPTCP is not notified what kind of path to select and the selected paths are not always appropriate to MPTCP.

On the other hand, to solve various problems of the current IP based architecture including the above problem, some new generation networking technologies have being studied. SDN(Software Defined Networking)[3] is one of the technologies that is drawing attention in the framework of the new generation network. In SDN, the forwarding process of packets is separated from the routing process. This can realize flexible communication by dynamically controlling equipment that only performs data transfer processing by software.

OpenFlow[4] is an implementation of SDN. The best feature of OpenFlow is that the control function which defines network functions is separated from the forwarding function which performs processes such as packet transfer. Traditionally, in order to manage a network equipment, network operators could only control a user interface in a software developed by a vendor of the equipment. Therefore, it was possible for nobody except the vendor to define the network requirement of network function. However, by using OpenFlow, it becomes possible for any users to program a control function freely.

OpenFlow consists of a controller and switches. A OpenFlow controller can perform routing based on the route information that is notified by OpenFlow switches in real time. For example, suppression of congestion can be expected by selecting routes so that congested routes are avoided. Inevitably, by utilizing OpenFlow, we can perform routing which is suitable for MPTCP without changing any IP mechanism.

In this research, we propose a method to support IP for MPTCP by routing using SDN for the purpose of improving QoS. Especially, we target QoS of Web service. Moreover, we implement our method and evaluate its effectiveness by experiment.

II. Proposal

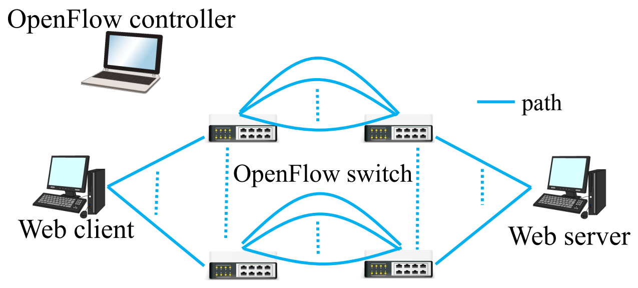

Fig. 1. An example of simple network

This method utilizes some OpenFlow switches and one OpenFlow controller between a Web client and a Web server. A path used by MPTCP diverges according to commands of the OpenFlow controller. Each OpenFlow switch assign packets over a MPTCP's path into diverged paths based on the traffic amount of them.

Let us explain an outline of our method with a simple network example shown in Fig. 1. In this network, the Web client and the Web server are connected with each other via some OpenFlow switches; they communicate to each other with MPTCP. There are some paths between pairs of OpenFlow switches. Each OpenFlow switch forwards incoming packets from/to a port which is determined according to the rules notified by the OpenFlow controller.

IP routing is assisted by switching among diverged paths between two OpenFlow switches to improve QoS of MPTCP. For the QoS improvement, the assignment of packets is performed so that loads between paths are balanced. To do this, each OpenFlow switch measures the amount of traffic of each path. When the amount of traffic of any path exceeds a threshold value, each switch sends packets to the path whose amount of traffic is the smallest.

III. Experiments

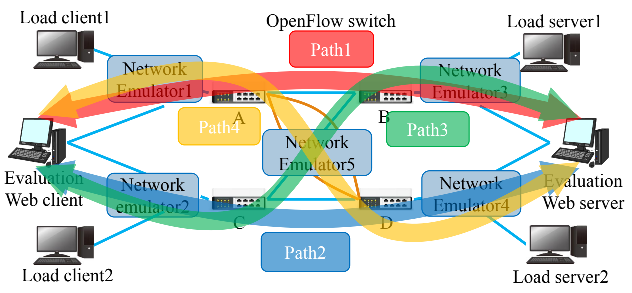

Fig. 2. The structure of the experiment

Our experimental environment is shown in Fig. 2. Specification of each machine are shown in Table I through Table VIII.

TABLE I
Specification of evaluation web client

Product	Dell Vostro 200
OS	Ubuntu 16.04
CPU	Intel(R) Core(TM)2 Duo CPU E7200@2.53GHz
memory	2.0GB
Congestion control	LIA
Path manager	full-mesh
Scheduler	round robin

TABLE II
Specification of evaluation web server

Product	Dell Vostro 230
OS	Ubuntu 16.04
CPU	Pentium(R)Dual-Core CPU E5700@3.00GHz
Memory	3.0GB
Congestion control	LIA
Path manager	full-mesh
Scheduler	round robin

TABLE III
Specification of OpenFlow controller

Product	Dell Latitude E6530
OS	Ubuntu 16.04
CPU	Intel(R)Core(TM)i7-3540M CPU@3.00GHz x4
Memory	4.0GB

TABLE IV
Specification of OpenFlow switch

Product	Raspberry Pi 3 Model B
OS	Rasbian Stretch 9.1
CPU	ARM Cortex-A53 (1.2GHz)
Memory	1GB

TABLE V
Specification of load client1

Product	Dell Vostro 270s
OS	Ubuntu 12.10
CPU	Intel(R) Pentium(R) CPU G2030@3.0GHz × 2
Memory	4.0GB

TABLE VI
Specification of load client2

Product	Dell Vostro 270s
OS	Ubuntu 12.04 LTS
CPU	Intel(R) Core(TM)i3-3240 CPU @3.40GHz × 4
Memory	4.0GB

TABLE VII
Specification of load server1

Product	Dell Vostro 270s
OS	Ubuntu 12.04 LTS
CPU	Intel(R) Core(TM)i3-3240 CPU @3.40GHz × 4
Memory	4.0GB

TABLE VIII
Specification of load server2

Product	Dell Vostro 270s
OS	Ubuntu 12.04 LTS
CPU	Intel(R) Core(TM)i3-3240 CPU @3.40GHz × 4
Memory	4.0GB

Under this environment, we make four paths between the Web client and the Web server. We refer to them as Path 1, Path 2, Path 3 and Path 4. MPTCP utilizes all of the four paths. In this experiment, as the first step of our research, we only use the proposed method for Path 4. For this reason, we consider the following two paths as candidates of Path 4.

- A path with load traffic; its round trip time is 0ms
- A path without load traffic; its round trip time is 100ms

We refer to the former and the latter as Path 4-1 and Path 4-2, respectively. Here we use the three experimental configurations

TABLE IX
COMMUNICATE ENVIRONMENT

		Path 1	Path 2	Path 3	Path 4-1	Path 4-2
Env1	delay	0ms	100ms	100ms	100ms	0ms
	load	O	X	O	X	O
Env2	delay	0ms	100ms	100ms	100ms	0ms
	load	O	X	X	X	O
Env3	delay	0ms	100ms	0ms	100ms	0ms
	load	O	X	O	X	O

shown in Table IX. These paths are composed of four network emulators(network emulator 1 through network emulator 4), which also works as a router. We use RaspberryPi3 and ryu[5] as OpenFlow switch and OpenFlow controller respectively. In order to eliminate an impact of controll traffic on experimental result, the OpenFlow controller and the each OpenFlow switch are connected on the network independent of the data transfer network. The load clients communicate with the corresponding load servers and generate Web traffic for load. In addition, we use four network emulators with function of router to generate delay on each path. Moreover, we use another network emulator on Path 4 to generate different delay between Path 4-1 and path 4-2. We use Dummynet[6] as a network emulator. We show the specification of the network emulators are shown in Table X through TableXIV.

TABLE X
SPECIFICATION OF NETWORK EMULATOR 1

Product	XCY J1800
OS	FreeBSD 11.1 RELEASE
CPU	Intel(R) Celeron(R) CPU 1037U@1.80GHz
Memory	2.6GB

TABLE XI
SPECIFICATION OF NETWORK EMULATOR 2

Product	XCY 1037U
OS	FreeBSD 11.1 RELEASE
CPU	Intel(R) Celeron(R) CPU J1800@2.41GHz
Memory	3.0GB

TABLE XII
SPECIFICATION OF NETWORK EMULATOR 3

Product	Dell Vostro 220s
OS	FreeBSD 11.1 RELEASE
CPU	Intel(R) Core(TM)2 Duo CPU E7200@2.53GHz
Memory	2.6GB

TABLE XIII
SPECIFICATION OF NETWORK EMULATOR 4

Product	Dell Vostro 220s
OS	FreeBSD 9.2 RELEASE
CPU	Intel(R) Core(TM)2 Duo CPU E7500@2.93GHz
Memory	4.0GB

TABLE XIV
SPECIFICATION OF NETWORK EMULATOR 5

Product	XCY 1037U
OS	FreeBSD 11.1 RELEASE
CPU	Intel(R) Celeron(R) CPU J1800@2.41GHz
Memory	3.0GB

We consider the mean throughput as QoS parameters. We compare results of the proposed method with those which treat either the Path 4-1 or Path 4-2 as Path 4.

IV. RESULTS

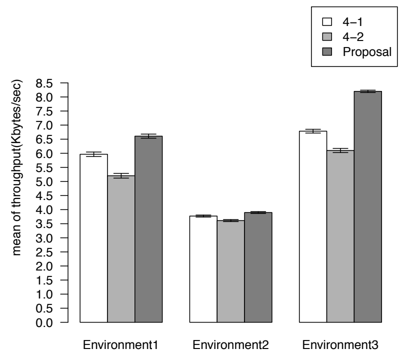

Fig. 3. Mean of throughput

Our experimental results are shown in Fig. 3. This shows the 95% confidence interval obtained by the *t*-test with the significance level 5%.

Figure 3 indicates that the mean throughput is improved by using the proposed method in all environments. Consequently, we confirm that by using the proposed method in Path 4, the load balancing using Path 4-1 and Path 4-2 was appropriately performed, and then the load of the Path 4 was suppressed. From the above results, we indicate that QoS was improved by applying our method in MPTCP communication under this environment. Therefore, we could confirm the effectiveness of our method.

V. CONCLUSIONS

In this research, we proposed a method for assisting IP routing for MPTCP by using SDN. We target Web services and implemented it. From the experimental results, the effectiveness of this method was confirmed.

ACKNOWLEDGMENT

This work was supported by MEXT KAKENHI Grant Number JP 16K00122.

REFERENCES

[1] A. Ford et al.,RFC6824,Jan. 2013.
[2] J. Postel. Internet Protocol. RFC 791 (INTERNET STANDARD), September 1981. Updated by RFCs 1349, 2474, 6864.
[3] https://www.opennetworking.org/sdn-resources/sdn-definition
[4] https://www.opennetworking.org/sdn-resources/openflow
[5] “ ryu ” . https://osrg.github.io/ryu/.
[6] “ Dummynet ” . http://info.iet.unipi.it/ luigi/dummynet/.