Journal of Information Processing Vol.25 924–933 (Oct. 2017)

[DOI: 10.2197/ipsjjip.25.924]

Regular Paper

A Multipath OpenFlow Controller for Multiple TCP Stream Applications

Che Huang[1,a) Chawanat Nakasan[1,b) Kohei Ichikawa[1,c) Yasuhiro Watashiba[1,d) Hajimu Iida[1,e)

Received: February 24, 2017, Accepted: April 17, 2017

Abstract: A large amount of data needs to be transferred from one site to another as fast as possible in the computational science fields. To achieve high-speed data transfer, many applications utilize multiple TCP streams. However, since multiple TCP streams of applications are usually routed according to the default IP routing protocol, only a single shortest path among the multiple paths can be utilized for the data transfer. This research proposes a multipath controller that increases the performance of data transfer by leveraging multiple paths simultaneously for parallel TCP streams. For this purpose, we utilize the Software-Defined Networking (SDN) technology and its implementation, OpenFlow. Furthermore, we proposed a method to determine optimal numbers of parallel TCP streams to be assigned for each path according to its own network condition. This paper presents the design and implementation of the proposed system. As a case study, we applied our proposed system on GridFTP and evaluated the performance improvement. The results demonstrate that our proposed system accelerates the data transfer of GridFTP in both a virtual and a real global-scale environment.

Keywords: multipath, Software-Defined Networking, OpenFlow, data transfer

1. Introduction

High-speed data transfer from one site to another is a necessary platform service for future Big Data and data-intensive science [5]. With the rapid development of computer technology, the amounts of data have been increased up to the petabyte and even exabyte scale in the computational science research fields. In addition, in international environments for collaborative research today, large-scale data is not stored only on one site; therefore, high-speed data transfer between sites is very important.

To achieve high-speed data transfer in widely-distributed environments, many applications utilize multiple TCP streams simultaneously to transfer data. Using multiple TCP streams in parallel can improve aggregate bandwidth over using a single TCP stream by mitigating the negative effects of packet loss and 'slow start' mechanism of TCP. There have been a number of proposed schemes designed for applications to use multiple TCP streams such as XFTP [3], GridFTP [2], MultiTCP [24], PATTHEL [4], Multipath TCP (MPTCP) [6], to increase the performance of data transfer.

On the other hand, there are usually multiple network paths (multipath) available between widely-distributed sites, however, these multiple paths are not efficiently utilized by applications. Because even if the applications use multiple TCP streams in parallel, those multiple TCP streams are basically routed according to the default IP routing protocol, and only a single shortest path among the multiple paths is used for the data transfer. The primary reason for this problem is that there is a gap between application demands and the network architecture, and the applications are unaware of the information on the network layer. Thus, there is still much room for improvement in data transfer by applying some traffic engineering technologies using different multiple paths simultaneously.

In this study, we propose a multipath controller that distributes the parallel TCP streams of applications into multiple network paths by utilizing Software-Defined Networking (SDN)-based traffic engineering techniques. SDN is a newly emerged concept that brings software programmability to networks and allows us to control routing assignment of the entire networks from the viewpoint of applications. Furthermore, to optimize the data transfer performance, we developed a prediction model to determine optimal numbers of parallel TCP streams to be assigned for each path according to its own network condition.

We have developed our system based on OpenFlow [18], which is standardized by the Open Networking Foundation (ONF) [20] and one of the most used standard protocols for SDN. We applied our proposed multipath controller to GridFTP as an actual case study to demonstrate the effectiveness and practicality of our proposed system, because GridFTP is one of the most common data transfer services using multiple TCP streams and it is used widely in the computational science research fields.

The rest of this paper is organized as follows. Section 2

1 Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
a) huangche@is.nasit.jp
b) chawanat.nakasan.cb5@is.nasit.jp
c) ichikawa@is.naist.jp
d) watashiba@is.naist.jp
e) iida@itc.naist.jp

describes related existing researches on high-speed data transfer using multipath. Section 3 explains our approach leveraging different multiple paths using the SDN technology and how we optimize the assignment of TCP streams in multipath routing. Section 4 describes the implementation of our system. Section 5 shows evaluation results of the proposed system. Section 6 presents the conclusion of this paper and future works.

2. Related Work

There have been a number of proposed schemes for providing multipath to applications [26], [27], [28]. Some of them have similar approach with our study. We describe several examples as follows.

Kissel et al. developed a new session layer protocol, called Phoebus, and made it available from the applications by taking advantage of the Dynamic Circuit Network (DCN), which is deployed on national academic research networks such as Internet2 [16]. Phoebus provides Phoebus Gateways (PGs), which hide different transport layer protocols such as UDT [8] and TCP behind the session layer, and offer an easy method to improve the throughput for general applications.

In addition, Dan et al. extended the research idea of Kissel et al. and proposed a system that improves the network throughput by assigning multiple TCP streams to different multiple paths brought on the conventional IP-based routing, the DCN provided by Phoebus and the OS^{3}E of Network Development and Deployment Initiative (NDDI) [9]. NDDI/OS^{3}E is a service that builds Layer 2 networks dynamically on their OpenFlow network infrastructure.

The research of Kissel et al. leveraged high-speed circuits brought by the DCN of Internet2 and made the multiple transport protocols available on the DCN. However, their method does not provide an interface for the users to control routing paths inside the Internet2. Once, a path is provided through the DCN, the provided path is just statically used for the multiple TCP streams of application. Dan et al. use a Layer 2 network dynamically built upon their OpenFlow network provided by NDDI/OS^{3}E. However, the users cannot each control individual routing inside the NDDI/OS^{3}E. The routes for the Layer 2 network is determined at the time it is created on the OpenFlow network. Therefore, Dan et al.'s system also just assigns the multiple routes provided by NDDI/OS^{3}E to the multiple TCP streams of application. Therefore, those two approaches are not flexible enough to optimize multiple paths between widely distributed sites.

However, recently, the optimization of the network layer based on the requests from the application layer has been gathering much attention with the development of the OpenFlow network. The Research Infrastructure for large-Scale network Experiments (RISE) service provided by Japan Gigabit Network eXtreme (JGN-X) is one of the services that allow users to control individual OpenFlow switches of the service [13], [15]. Considering the recent trend of the research, it is necessary to optimize multipath assignment by controlling the individual switches under an environment where OpenFlow switches are available for end-to-end communication.

Although the above existing researches tried to generate multiple TCP streams at the application level, a method generating multiple TCP streams at the system level, Multipath TCP (MPTCP), has also been proposed recently. There is also a research [25] that combines MPTCP and OpenFlow to achieve high-speed data transfer. However, the research also did not consider an environment where OpenFlow switches are available for end-to-end communication. In terms of routing multiple TCP streams, the difference between multiple streams of application level and multiple streams of MPTCP does not matter essentially. In this study, we evaluate our proposed method with application level multiple streams using GridFTP, because GridFTP has already been widely used for multiple TCP streams while MPTCP is not available widely.

3. Approach and Design

In this study, we proposed a multipath controller that improves the data transfer performance by assigning parallel TCP streams of an application to a number of different paths in the environment where OpenFlow switches are available for end-to-end communication. By aggregating the available bandwidth from multiple different paths, the performance of data transfer would be improved drastically.

However, to achieve the best performance using multiple paths, the strategy of how many TCP streams should be assigned for each path is also an important factor. The simplest method is to create many enough TCP streams and distribute these TCP streams equally over the multiple paths. However, this is obviously inefficient in terms of resource usage. Therefore, it is necessary to figure out the optimal combination of multipath and the number of parallel TCP streams.

We therefore tried to develop a method to determine optimal numbers of parallel TCP streams to be assigned for each path according to its own network condition. There are many factors affecting the transfer speed in a network connection; the two major factors are the bandwidth and the latency. Since each network path has different bandwidth and latency, the optimal number of parallel TCP streams to get the best performance may be different for each path. In order to develop a prediction model to determine optimal numbers of TCP streams, we figured out the relationship between the optimal number of TCP streams and network conditions.

3.1 Multipath Controller

Figure 1 shows our proposed parallel transfer of applications in an OpenFlow network. We aggregate multiple routes to increase the available bandwidth by assigning each of multiple TCP streams to different routes. In this study, we construct an OpenFlow controller that dynamically calculates available paths using breadth-first search between sites and allocates the calculated routes for applications.

We have designed general interfaces to request multiple routes so that any applications can request multiple routes for their data transfer. The following two functionalities are designed for the purpose: 1) Searching the specified number of paths between sites and setting aside the found paths for later use, 2) Installing appropriate flow entries into OpenFlow switches in response to

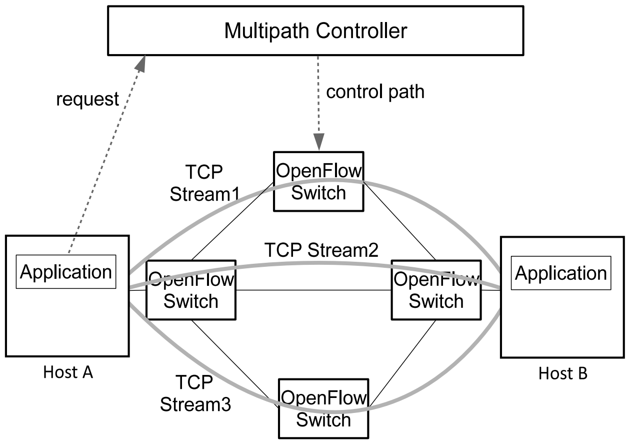


Fig. 1 The concept of multiple TCP streams assignment in the OpenFlow network.

the request of TCP connection from applications. The reason why we have two separated functions is that the TCP port numbers used for a communication are not determined until just before the TCP connection is opened. To install a flow entry into an OpenFlow switch, we need information on a set of source and destination addresses and TCP port numbers. Therefore, we search multiple available paths between sites first when the source and the destination addresses are provided. And then, we create actual flow entries when TCP port numbers are determined. Thus, we designed two separated functions to find multiple available routes and assign the routes for each actual TCP connection.

3.2 Prediction Model to Optimize TCP Stream Assignment in Multipath Routing

There are several existing researches to predict the maximum network throughput with multiple TCP streams, however, few researches have been conducted to find the optimal number of TCP streams that can achieve the maximum throughput. We therefore derived the prediction model for an optimal number of TCP streams based on some existing models for the network throughput.

According to Hacker et al Model [10], when an application opens a single stream, the maximum network throughput can be represented as:

$$Th <= \frac{MSS}{R}\frac{c_0}{\sqrt{p}}. \tag{1}$$

Th represents the maximum throughput, MSS is the maximum segment size of TCP, R is the round trip time, p is the packet loss rate and c_0 is a constant.

Hacker et al. [10] also claim that the aggregated throughput of parallel streams can be calculated with the throughput of a single stream multiplied by the number of streams. In addition, Dinda et al. [17] show p would increase as the number of parallel streams increases and the network gets congested. Therefore, Eq. (1) can be rearranged for n streams as:

$$Th_n <= \frac{MSS \times c_0}{R}\left(\frac{n}{\sqrt{p_n}}\right). \tag{2}$$

n represents the number of parallel streams, p_n is the packet loss rate when n streams are used on the network. If we use too many streams, p_n increases dramatically and Th_n decreases.

According to Ref. [14], the packet loss rate in a network, p_n, is determined only by the available bandwidth(B)-latency(R) product per TCP connection (i.e., BR/n) and can be represented as:

$$p_n = \left(c_1\left(\frac{BR}{n}\right)^2 + c_2\frac{BR}{n} - c_3\right)^{-1}. \tag{3}$$

c_1, c_2 and c_3 are constant and positive numbers.

After placing p_n in Eq. (2), the total achievable throughput Th_n is calculated as follow:

$$Th_n <= \frac{MSS \times c_0}{R}\left(\frac{n}{\sqrt{\left(c_1\left(\frac{BR}{n}\right)^2 + c_2\frac{BR}{n} - c_3\right)^{-1}}}\right). \tag{4}$$

Since Eq. (4) is a convex upward function, we can get an optimal n that maximizes the Th_n by solving the following partial differential equation for n:

$$\frac{\partial Th_n}{\partial n} = 0. \tag{5}$$

If we assume MSS is a relatively static value, the solution of Eq. (5) is given by the following equation:

$$n = \frac{c_1}{2c_3}BR. \tag{6}$$

Since c_1 and c_3 are constants, Eq. (6) can be simplified as follows with a single constant value, a:

$$n = aBR. \tag{7}$$

Since n is actually the number of TCP streams, it should be equal to or greater than 1. The equation can be therefore written as:

$$n = \max(1, aBR). \tag{8}$$

This result seems to be too simplified. However, we got this result by just combining the existing known models for the maximum aggregated bandwidth and the packet loss rate, as calculated in the above. Also, this result matches intuitive expectations. If we have a larger bandwidth-delay product, the number of optimal parallel TCP streams will increase accordingly.

The constant values, from c_1 to c_3, which define the packet loss rate, are determined by the characteristics of the network we use. Therefore, the constant value, a, in Eq. (8) will be also determined by the characteristics of the network. To determine the value, a, we need to measure several combinations of n, B and R. We will calculate the actual value of a later in the evaluation section.

4. Implementation

We developed our multipath controller using Trema [22], a framework for developing OpenFlow controllers in Ruby and C. We developed our controller based on the *routing_switch controller* [19] included in Trema Apps [23]. Trema Apps is a sample application set for Trema. *Routing_switch controller* is a simple OpenFlow controller that calculates the shortest-hop path between hosts using Dijkstra's algorithm and installs flow entries into the OpenFlow switches for the path. Our controller utilizes this default shortest-hop routing for the normal communications that do not request multipath routing.

We have implemented the two functionalities mentioned in the previous section, 1) Searching the specified number of paths between sites, and 2) Installing flow entries into OpenFlow switches, as XML-RPC interfaces on our OpenFlow controller. By implementing the functions as generic XML-RPC interfaces, any application can access our proposed controller. In our implementation, we named these two interfaces as *AssignMultipath* and *MakePath* respectively.

In order to adapt GridFTP to our proposed system, we implemented the required functionality to support our proposed system as one of the Globus XIO communication drivers [1] that GridFTP uses as a communication library. Globus XIO is a plug-in framework of I/O libraries that is implemented in the Globus Toolkit [7]. Globus XIO allows applications to support various protocols and file formats by implementing plug-ins for each different communication method and file and storage access. In this study, we created a new communication driver based on a standard XIO TCP driver which is one of the built-in default drivers in Globus XIO. Our communication driver establishes TCP connections in collaboration with our multipath controller, so that each TCP connection can take different paths.

To use our proposed controller, users need to go through the following two-step process. The first step is acquiring available multiple paths using *AssignMultipath*; the second step is deployment of actual flow entries using *MakePath*. Since the first step for acquiring multiple paths is a preprocess step before actual communication starts, we created a lightweight client program called *assign_mpath* just for calling the *AssignMultipath* interface. Also, we implemented an XIO driver to access the *MakePath* interface for the second step, and implemented the XIO driver to be called from GridFTP.

The actual execution procedure of program is as follows:

```
> ./assign_mpath <controller_address>
<port> <src_ip> <src_mac> <dst_ip>
 <path_num>

> MPATH_ASSIGNMENT_ID=<id> globus-url-
copy -p <path_num> test.data gsiftp:
//Host $B$/test.data
```

assign_mpath program requires the IP address of the OpenFlow controller (`controller_address`), the port number of the controller (`port`), the source IP address (`src_ip`), the source MAC address (`src_mac`), the destination IP address (`dst_ip`) and the number of paths needed to be assigned (`path_num`). *assign_mpath* outputs the number of assigned paths and an assignment ID. This assignment ID is a unique ID identifying the assigned paths and will be used later for calling the *MakePath* interface. *globus-url-copy* is a GridFTP client provided by the Globus Toolkit. In order to give the assignment ID to our developed XIO driver, we use an environment variable, `MPATH_ASSIGNMENT_ID`. *globus-url-copy* is therefore launched with the variable, `MPATH_ASSIGNMENT_ID`.

4.1 How the System Works

Figure 2 illustrates the proposed system. The proposed system

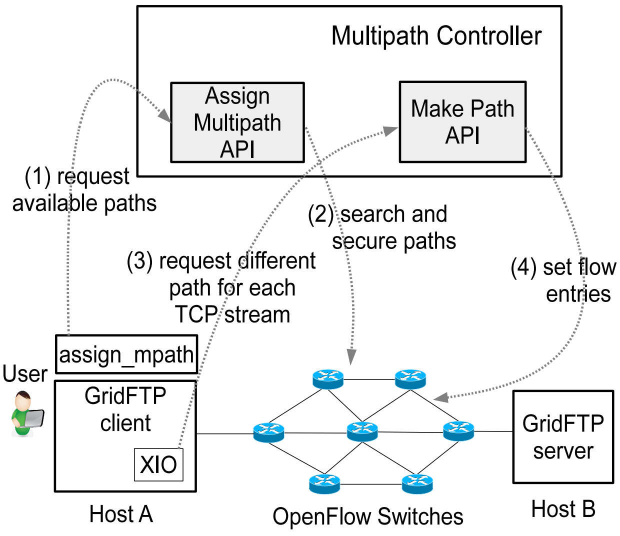


Fig. 2 Overview of our proposed multipath controller and GridFTP.

performs the following steps:

- First, in (1), a user launches the *assign_mpath* program. It accesses the *AssignMultipath* interface implemented on our OpenFlow controller, and requests available paths from Host *A* to Host *B*.
- Next, in (2), the *AssignMultipath* calculates the available paths with the breadth-first search algorithm from Host *A* to Host *B* based on the topology of the OpenFlow network. The *AssignMultipath* secures the specified number of available paths and returns the number of paths with an assignment ID to the user. The assignment ID will be used to refer to the assigned paths for later use.
- Then, the user starts the GridFTP client with the assignment ID obtained in the previous step.
- In (3), our implemented XIO driver loads the assignment ID. Then, the XIO driver accesses the *MakePath* interface on the OpenFlow controller and requests to create an individual path each time the GridFTP client opens a TCP stream via the XIO driver.
- In (4), the *MakePath* finds the set of paths assigned by the *AssignMultipath* using the assignment ID and acquires a path from the path set, and then installs flow entries into the OpenFlow switches for the requested TCP stream. The *MakePath* creates flow entries using the source and destination IP addresses and the source TCP port number as match conditions, and installs the flow entries to each OpenFlow switch for the path.
- Finally, the requested TCP stream starts the communication according to the assigned route. Steps (3) and (4) are applied repeatedly for the subsequent TCP streams.

5. Evaluation

To verify the effectiveness of the proposed system, we performed evaluations comparing the performance of the GridFTP with/without our proposed method. For retrieving the best possible performance, we conducted the evaluations on a virtual environment first. We then also performed some evaluations on a real global-scale environment to evaluate the practicality of our

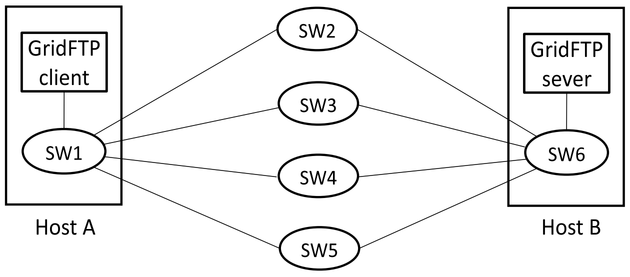


Fig. 3 Topology *A* on a local testbed (the bandwidth and the latency are configured to 100 Mbps and 0 ms on each path).

proposal.

5.1 Experiments Using a Virtual Environment

In our virtual environment, we conducted our experiments over a simple topology to confirm if our proposed system can achieve the expected results. Furthermore, we designed another more realistic topology, which has some overlapped links among different paths, to verify the effectiveness of our system.

5.1.1 Experimental Environment

Our experimental virtual environment is constructed on six virtual machines deployed on each of six physical VMware ESX machines equipped with two Intel Xeon E5649 processors and 48 GB memory. We assigned two virtual cores and 2 GB memory for each virtual machine, and setup CentOS 6.5 on each of them. These virtual machines share a single 1 Gbps network switch physically, and the actual available bandwidth is about 941 Mbps measured by iperf between two hosts. In this study, in order to see the effect using multipath, we have limited each link performance up to 100 Mbps, so as not to exceed the physical performance limitation.

We installed the GridFTP on two machines: Host *A* and Host *B*, and used these two machines as a client and a server respectively. In this experiment, due to the lack of hardware OpenFlow switch resources, we prepared multiple OpenFlow switches by installing a software-based implementation of OpenFlow switch, Open vSwitch [21]. We deployed several Open vSwitches between Host *A* and *B*, and constructed an OpenFlow network which has multiple paths between Host *A* and *B*.

In addition, to make the communication of GridFTP pass through the OpenFlow network, we also installed Open vSwitch on Host *A* and *B*. The Open vSwitches on Host *A*, Host *B* and other Open vSwitches are connected by GRE links, which is an IP-based point-to-point tunneling protocol. By changing the combination of the GRE links, we can easily construct various topologies for the experiments and configure each path with different bandwidth and latency.

In this virtual environment, we conducted the experiments on two network topologies, topology *A* and *B*. Topology *A* (**Fig. 3**) is a simple topology which has four independent paths, and the bandwidth and latency of each path are configured to 100 Mbps and 0 ms respectively. Topology *B* (**Fig. 4**) assumes a more realistic situation. It has some overlapped links and the bandwidth and latency are configured as shown in Fig. 4. In the figures, SW1 to SW6 denote Open vSwitches.

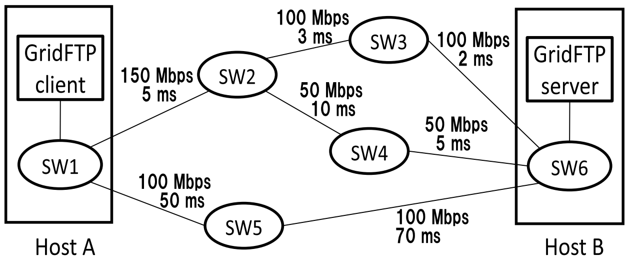


Fig. 4 Topology *B* on a local testbed (different bandwidth and latency are configured for each path as shown in the Figure).

Latency (ms)	Number of TCP streams							
	1	2	3	4	5	6	7	8
0	**374.66**	374.77	374.66	374.67	374.89	-	-	-
20	376.10	**375.02**	375.45	375.55	375.44	375.74	375.89	-
40	387.38	376.76	**376.30**	376.36	376.75	376.40	376.65	376.98
60	390.40	385.32	378.94	**377.10**	377.47	377.58	377.26	377.48
80	391.03	384.41	381.96	379.30	**378.09**	378.43	378.14	378.43
100	393.63	387.84	384.26	382.46	381.21	379.72	**378.44**	378.91

Fig. 5 Transfer time (sec) of 2 GB file with 50 Mbps link.

5.1.2 Determine the Parameter of Prediction Model

In order to determine the constant value, *a*, of Eq. (8) in our virtual environment, we have measured the optimal number of TCP streams with various network conditions changing the available bandwidth and latency. We utilized two virtual machines connected with a single 1 Gbps network switch. Traffic control tool (tc) of Linux [11] was used to configure the available bandwidth and latency between the two virtual machines. We have measured the time needed to transfer a file of 2 GB from one host to another host under different conditions where the available bandwidth is limited to 50 Mbps or 100 Mbps with changing the added latency from 0 to 140 ms.

Figure 5 shows the part of the observed data in the case where the available bandwidth is limited to 50 Mbps. We have repeated this measurement 12 times for each case. The data presented in Fig. 5 reflects the average of 10 trials excluding the highest and lowest ones. In the Figure, the best results for each different latency are highlighted with red. For example, we can see that using 4 TCP streams achieved the best performance where the latency is configured to 60 ms.

Using the measurement results, we can calculate the constant value, *a*. From the result of the average value calculated with the measurement results, we determined that *a* is about 0.001495 in our virtual environment.

Figure 6 plots the measured optimal numbers of TCP streams and the predicted lines based on our proposed model. We can see that the measured values are very close to our prediction results. Therefore, by using Eq. (8), we can calculate the optimal number of TCP streams if the available bandwidth and latency are given.

5.1.3 Results of Experiments

In the experiments, the data transfer time was measured. In our proposed system, it is necessary to run the *assign_mpath* command in advance to find the routes. But, we did not measure the time taken for *assign_mpath*, since these experimental topologies are very small and the required time for executing *assign_mpath* is very short. We leave such evaluation on the scalability of *as-*

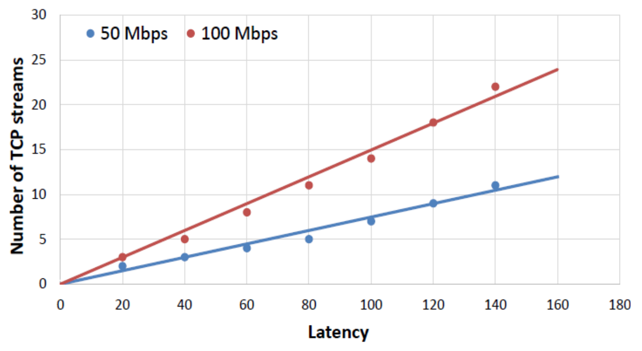


Fig. 6 Relationship between the optimal number of TCP streams and the latency.

Table 1 Comparison of transfer time in topology *A*.

Number of streams	Proposed multipath system	Conventional single path method
4	23.40 sec	90.53 sec

sign_mpath with larger and more complex topologies as a future issue.

In addition, we measured the used bandwidth by periodically monitoring packet counters on the OpenFlow switches. Each packet counter on OpenFlow switches records the number of transferred packets and transferred bytes for a flow-basis. Thus, this information is useful to measure the bandwidth of each TCP stream of the GridFTP separately. We measured the counter in the Open vSwtich on Host *B*, which is the destination of the data transfer.

In topology *A*, we conducted the experiment by transferring a file of 1 Gbyte. In this topology, there are four available paths, 1) $SW1$-$SW2$-$SW6$, 2) $SW1$-$SW3$-$SW6$, 3) $SW1$-$SW4$-$SW6$ and 4) $SW1$-$SW5$-$SW6$. In this paper, $SW1$-$SW2$-$SW6$ represents a path which walks through switches $SW1$, $SW2$, $SW6$ in this order. The optimal number of TCP streams is calculated as 1 for each path in this topology, because the added latency for each path is configured to 0 ms.

Table 1 shows the experimental results of topology *A*. The result shows that our proposed multipath system successfully distributes four multiple TCP streams of GridFTP into each different path and improved data transfer speed, while the conventional single path method just uses a single path for all four TCP streams. In the case using four parallel TCP streams, the proposed system achieved around four times better performance compared to the conventional single path method. **Figure 7** shows the used bandwidth summarized as stacked area graphs which are calculated from the average transferred bytes per second. We can see that the performance of each TCP stream is also very stable.

In topology *B*, we compared the proposed optimal assignment and a simple round robin assignment by transferring a file of 10 Gbyte to evaluate the efficiency of the proposed optimal assignment method. There are three available paths, 1) $SW1$-$SW2$-$SW3$-$SW6$, 2) $SW1$-$SW2$-$SW4$-$SW6$ and 3) $SW1$-$SW5$-$SW6$ in the topology. With the proposed optimal assignment, the assignment of TCP streams for each path are decided based on the calculated optimal number of TCP streams of each path. On the other hand, with the simple round robin assignment, the assignment of TCP streams for each path are equally distributed. In

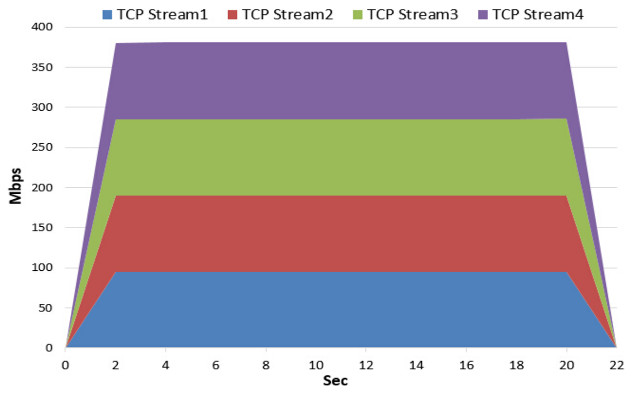


Fig. 7 Used bandwidth of each TCP stream in topology *A* with four parallel TCP streams.

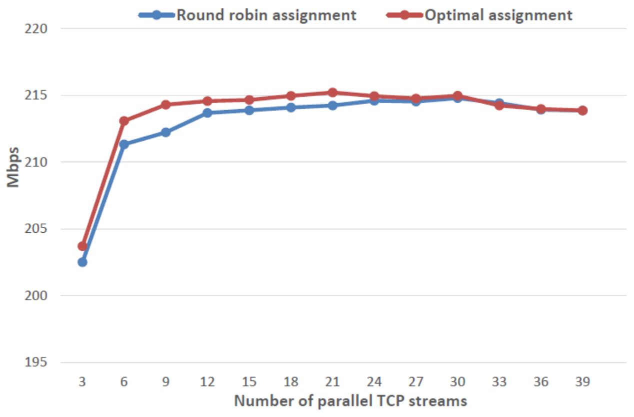


Fig. 8 Comparison of the average data transfer speed between the optimal assignment and the round robin assignment in topology *B*.

the optimal assignment method, the optimal assignment of TCP streams is calculated as that 2 streams for each first and second path and 18 streams for the third path. Therefore, using 22 TCP streams in total is the optimal number of streams in this case.

To compare the performance between our optimal method and the simple round robin method, we actually measured the transfer time with increasing the number of TCP streams from 3 to 39 by 3. Since the optimal assignment of streams is 2, 2 and 18 streams for each of three paths, we assigned the TCP streams to the three paths with a ratio of 1:1:9 in the proposed method. On the other hand, we assigned the TCP streams in a round robin manner for the simple round robin method. We repeated the same experiment 10 times and calculated the average for the results.

Figure 8 shows the average data transfer speed of the optimal assignment and the round robin assignment with increasing the number of parallel TCP streams. As shown in Fig. 8, the proposed optimal assignment method achieves an overall better throughput than the round robin assignment. This also means that the performance of the optimal assignment method is converged to the peak performance more quickly. Also, the peak of the performance is located in the position where the number of TCP streams is around the predicted optimal number, 22. The results demonstrated the effectiveness of our proposed system with the proposed optimal TCP stream assignment.

5.2 Experiments using a Real Global-scale Environment

To evaluate the practicality of our method, we have also con-

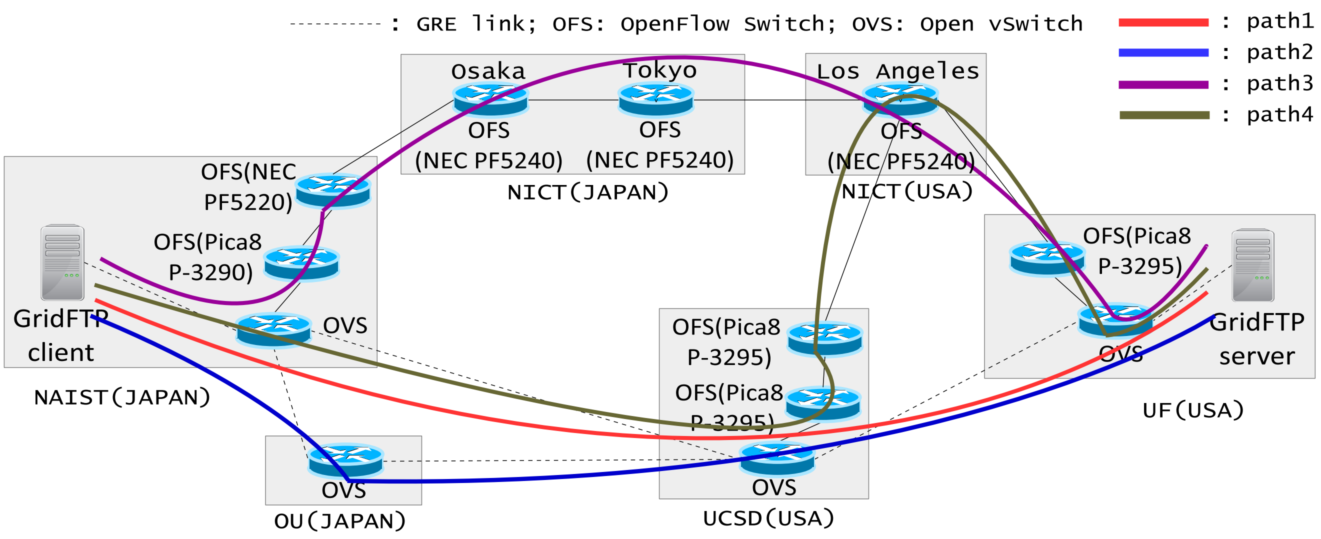


Fig. 9 Overview of the real global-scale experimental environment.

ducted some experiments in a real global-scale environment.

5.2.1 Experimental Environment

For the evaluation, we used the resources provided by PRAGMA Experimental Network Testbed (PRAGMA-ENT) [12]. PRAGMA-ENT provides a large scale OpenFlow network composed of computing resources and international academic networks. The part of resources is also connected through GRE over the public Internet as alternative paths. We used a part of the resources provided by PRAGMA-ENT.

Figure 9 shows the overview of the experimental environment. We installed GridFTP on a virtual machine as a client, and deployed an Open vSwitch and two hardware OpenFlow Switches (Pica8 P-3290 and NEC PF5220) at Nara Institute of Science and Technology, Japan (NAIST). Also, we installed GridFTP on a virtual machine as a server, and deployed an Open vSwitch and a hardware OpenFlow Switch (Pica8 P-3290) at the University of Florida, USA (UF).

There are three sites between NAIST and UF: 1) National Institute of Information and Communications Technology, Japan (NICT) that deployed three hardware OpenFlow Switches (NEC PF5240): one is in Osaka Data Center, one is in Tokyo Data Center and another one is in Los Angeles Data Center, 2) Osaka University, Japan (OU) that deployed an Open vSwitch, 3) University of California, San Diego, USA (UCSD) that deployed two hardware OpenFlow Switches (Pica8 P-3290) and an Open vSwitch.

The experimental environment uses different international and academic networks and GRE: 1) The GRE connection between NAIST and UCSD is established over the TransPAC3 network; 2) The GRE connections between OU and NAIST, OU and UCSD are established over Science Information NETwork, Japan (SINET5); 3) NAIST, Osaka, Tokyo and Los Angeles Data Center of NICT are connected with RISE service over JGN-X; 4) The links between UF and Los Angeles, UCSD and Los Angeles, UCSD and UF are connected via Internet2 and California Research and Education Network (CalREN). All experiments on the global environments are conducted during weekends to reduce the impact from the background traffic, because there is usually a larger traffic during working days and a smaller traffic during weekends on these national research and education networks.

Table 2 Experiment result to determine value a in the global-scale experimental environment.

Path number	1	2	3	4
Latency (ms)	191	183	211	191
Available average bandwidth (Mbps)	728.4	638	916	780.5
S.D. of available bandwidth	0.78	3.82	4.97	20.43
Observed optimal number of streams	8	10	10	28
Predicted optimal number of streams	9	8	13	10

5.2.2 Determine the Parameter of Prediction Model

In order to determine the value, a, of Eq. (8) in our global-scale environment, we have also measured the available bandwidth and latency on several paths and figured out the optimal number of TCP streams for those paths. Figure 9 shows four shorter paths (path1 to 4) that our system found. Since longer paths than these four paths have too many overlapped links with the other paths and are not useful to evaluate, we use the four paths illustrated in Fig. 9.

Table 2 indicates the measurement results of the four paths. As the fourth path has a bigger standard deviation and the performance is not stable on the path, we use the other three paths to calculate the value, a. From the measurement results, we have determined the value, a as about 0.000065 in our international environment. Table 2 also shows the predicted optimal number of TCP streams. We can see that the numbers are slightly different from the observed optimal numbers, but still similar.

5.2.3 Results of Experiments

In the experiments, we compared our proposed system with two other methods, 1) the single path assignment method which is a conventional routing method just using a single path for multiple TCP streams, 2) the round robin assignment method which uses available multiple paths in a round robin manner. Our method uses available multiple paths based on the rate from the predicted optimal numbers of TCP streams for each path. For the evaluation, we transferred a file of 6 GB from NAIST to UF, and measured the transfer time and also measured the used bandwidth during the transfer. The measurement method is the same as the method used in our virtual environment experiments.

Since the used four paths have shared links each other, the

Table 3 Optimal assignment in the global-scale experimental environment.

Path number	1	2	3	4
Latency (ms)	191	183	211	191
Expected available bandwidth (Mbps)	660	200	740	170
Optimal number of streams	8	2	10	2

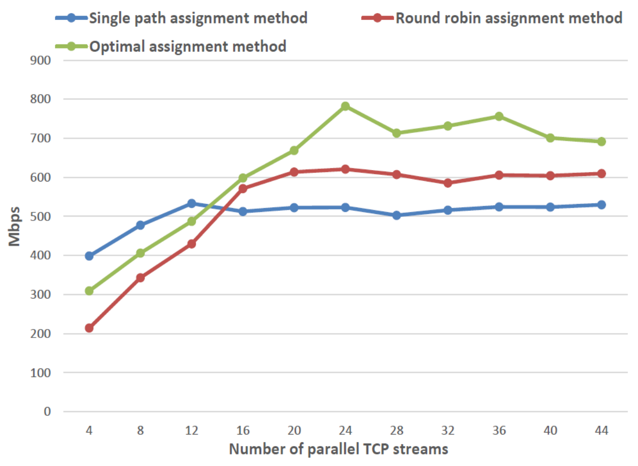


Fig. 10 Comparison of the average data transfer speed between the single path assignment, the round robin assignment and the optimal assignment.

available bandwidth would be reduced when these four paths are used simultaneously. We measured a standalone performance of each link and expected the available bandwidth of each path as shown in **Table 3**. Based on the expected bandwidth, we have calculated the optimal number of TCP streams for each path as 8, 2, 10 and 2 respectively. Therefore, we assigned TCP streams to the four paths with a ratio of 4:1:5:1 in our optimal assignment method.

Figure 10 shows the average speed of the data transfer while increasing the number of parallel TCP streams. From the results, in the case of using 4, 8 and 12 parallel TCP streams, the average speeds of the single path assignment method are better than the optimal assignment and the round robin assignment method. This is because our proposed system used path1, 2, 3 and 4 simultaneously, and only a few TCP streams were assigned for each path in the case of using smaller streams. Therefore, those TCP streams could not overcome the performance degradation of TCP's slow start mechanism. On the other hand, in the case of using the single path assignment method, all streams were assigned to the shortest path, path1, and achieved better performance than our method. However, when we used more than 16 streams, the optimal assignment and the round robin assignment method achieved better performance than the single path assignment. Especially, the maximum performance of our optimal assignment method reaches approximately 30% better than the round robin assignment method and approximately 60% better than the single path assignment method.

Figure 11 shows the used bandwidth for the round robin assignment method with 24 parallel TCP streams. The performance for path4 is slightly worse than the other paths because path4 is the longest and unstable path. The results show that the bandwidth keeps at around 880 Mbps and also achieved 900 Mbps as its best performance.

Figure 12 shows the used bandwidth for the optimal assignment method with 22 parallel TCP streams. The performance of path2 and 3 is worse than that of the other paths. But, the overall performance of the aggregated bandwidth is larger than the round robin assignment method. The results show that the traffic is stable and the bandwidth keeps at around 960 Mbps. In this experiment, our virtual machine host was equipped only with a 1 Gbps NIC. So, this result indicates that our proposed method achieved the performance that is close to the physical limitation of the hardware.

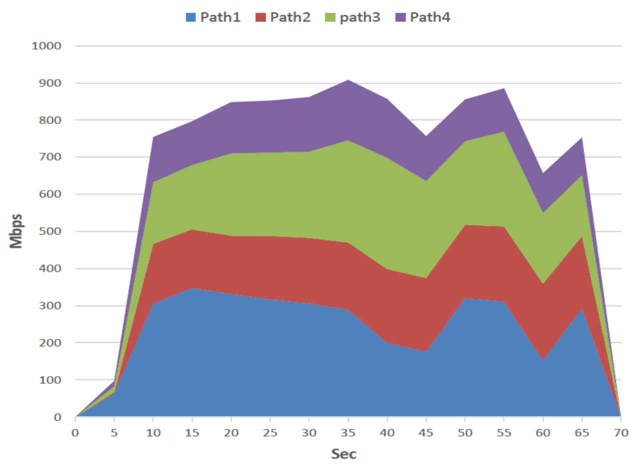


Fig. 11 Used bandwidth for the round robin assignment method in case of using 24 parallel TCP streams (6 parallel TCP streams are assigned for each path).

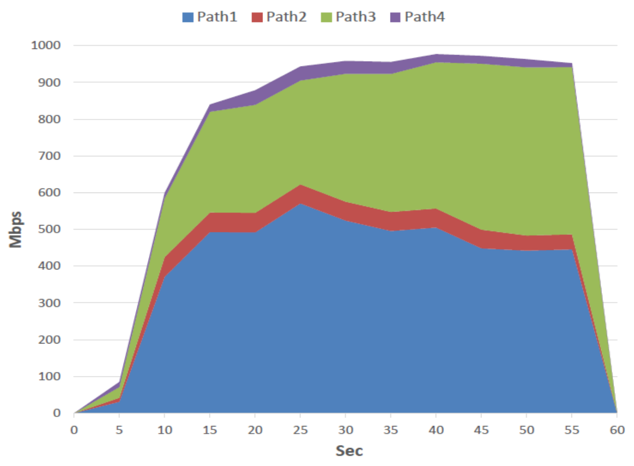


Fig. 12 Used bandwidth for the optimal assignment method in case of using 22 parallel TCP streams (TCP streams are assigned for each path with a ratio of 4:1:5:1 in order).

6. Conclusion

In this paper, we proposed a multipath controller providing multiple paths to the multiple TCP streams applications by using SDN technology based on OpenFlow. We also proposed a prediction model for optimal assignment of TCP streams. To demonstrate the effectiveness of our proposed controller, we adapted GridFTP into our proposed system as a case study and performed experiments using a virtual environment and a real global-scale environment. Experimental results showed that the proposed system improves the bandwidth usage and shortens the time of the GridFTP transfer.

For the future works, there are still some issues needing to be addressed. First, in our current OpenFlow controller, to simplify the design, we used a common parameter *a* of prediction model for all paths. Actually, for each network path, the performance could be improved by adjusting the parameter *a* according to network path conditions. Therefore, we plan to adapt the parameter adjustment function to improve the OpenFlow controller and conduct further experiments. Second, we just used simple topologies for the evaluations. To evaluate the practicality and performance of our OpenFlow controller, we need to test the controller with larger and more realistic networks. Benchmarking with simulated environments would be also considered. Third, we have not taken into account the use of multiple users. Currently, our system only focuses on optimizing the data transfer performance between two endpoints. We may need to optimize the TCP streams assignment in terms of the fairness among multiple users of the network.

Acknowledgments This work was partly supported by JSPS KAKENHI Grant Number 15K00170. The first author also gratefully expresses his gratitude to the Japan Ministry of Education, Culture, Sports, Science and Technology (MEXT) for scholarship.

References

[1] Allcock, W., Bresnahan, J., Kettimuthu, R. and Link, J.: The globus extensible input/output system (XIO): A protocol independent IO system for the grid, *Joint Workshop on HighPerformance Grid Computing and High-Level Paral-lel Programming Models in conjunction with International Parallel and Distributed Processing Symposium*, IEEE (2005).

[2] Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I. and Foster, I.: The Globus striped GridFTP framework and server, *Proc. 2005 ACM/IEEE Conference on Supercomputing*, p.54, IEEE Computer Society (2005).

[3] Allman, M., Kruse, H. and Ostermann, S.: An application-level solution to TCP's satellite inefficiencies, *Proc. 1st International Workshop on Satellite-based Information Services* (*WOSBIS*), Citeseer (1996).

[4] Baldini, A., De Carli, L. and Risso, F.: Increasing performances of TCP data transfers through multiple parallel connections, *IEEE Symposium on Computers and Communications, 2009, ISCC 2009*, pp.630–636, IEEE (2009).

[5] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S.: The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, *Journal of Network and Computer Applications*, Vol.23, No3, pp.187–200 (2000).

[6] Ford, A., Raiciu, C., Handley, M., Barre, S. and Iyengar, J.: Architectural guidelines for multipath TCP development, *RFC 6182* (2011).

[7] Foster, I., Kesselman, C. and Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations, *International Journal of High Performance Computing Applications*, Vol.15, No.3, pp.200–222 (2001).

[8] Gu, Y. and Grossman, R.L.: UDT: UDP-based data transfer for high-speed wide area networks, *Computer Networks*, Vol.51, No.7, pp.1777–1799 (2007).

[9] Gunter, D., Kettimuthu, R., Kissel, E., Swany, M., Yi, J. and Zurawski, J.: Exploiting Network Parallelism for Improving Data Transfer Performance, *High Performance Computing, Networking, Storage and Analysis* (*SCC*), *2012 SC Companion:*, pp.1600–1606, IEEE (2012).

[10] Hacker, T.J., Athey, B.D. and Noble, B.: The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network, *IEEE International Symposium on Parallel and Distributed Processing* (*IPDPS02*), pp.434–443, IEEE (2002).

[11] Hubert, B. et al.: Linux advanced routing & traffic control HOWTO (2009), available from ⟨http://www.lartc.org/howto/⟩.

[12] Ichikawa, K., Tsugawa, M., Haga, J., Yamanaka, H., Liu, T.-L., Kido, Y., U-Chupala, P., Huang, C., Nakasan, C., Chang, J.-Y., Ku, L.-C., Tsai, W.-F., Date, S., Shimojo, S., Papadopoulos, P. and Fortes, J.: PRAGMA-ENT: Exposing SDN Concepts to Domain Scientists in the Pacific Rim, *PRAGMA Workshop on International Clouds for Data Science* (*PRAGMA-ICDS 2015*) (2015).

[13] Ishii, S., Kawai, E., Kanaumi, Y., Saito, S.-i., Takata, T., Kobayashi, K. and Shimojo, S.: A study on designing OpenFlow controller RISE 3.0, *2013 19th IEEE International Conference on Networks* (*ICON*), pp.1–5, IEEE (2013).

[14] Ito, T., Ohsaki, H. and Imase, M.: On parameter tuning of data transfer protocol GridFTP for wide-area grid computing, *2nd International Conference on Broadband Networks, 2005*, pp.1338–1344, IEEE (2005).

[15] Kanaumi, Y., Saito, S.-I., Kawai, E., Ishii, S., Kobayashi, K. and Shimojo, S.: RISE: A Wide-Area Hybrid OpenFlow Network Testbed, *Ieice Trans. Comm.*, Vol.96, No.1, pp.108–118 (2013).

[16] Kissel, E., Swany, M. and Brown, A.: Improving GridFTP performance using the Phoebus session layer, *Proc. Conference on High Performance Computing Networking, Storage and Analysis*, p.34, ACM (2009).

[17] Lu, D., Qiao, Y., Dinda, P.A. and Bustamante, F.E.: Modeling and taming parallel TCP on the wide area network, *19th IEEE International Parallel and Distributed Processing Symposium*, Vol.1, p.68b, IEEE (2005).

[18] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J.: OpenFlow: Enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, Vol.38, No.2, pp.69–74 (2008).

[19] NEC Corporation: Routing switch, Trema App [Online], available from ⟨https://github.com/trema/apps/tree/master/routing_switch⟩ (accessed 2016-07-22).

[20] Open Networking Foundation [Online], available from ⟨https://www.opennetworking.org/⟩ (accessed 2016-07-22).

[21] Open vSwitch [Online], available from ⟨http://openvswitch.org⟩ (accessed 2016-07-22).

[22] Shimonishi, H., Takamiya, Y., Chiba, Y., Sugyo, K., Hatano, Y., Sonoda, K., Suzuki, K., Kotani, D. and Akiyoshi, I.: Programmable network using OpenFlow for network researches and experiments, *Proc. 6th International Conference on Mobile Computing and Ubiquitous Networking* (*ICMU 2012*), pp.164–171 (2012).

[23] Trema App [Online], available from ⟨https://github.com/trema/apps⟩ (accessed 2016-07-22).

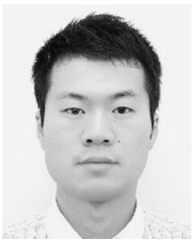
[24] Tullimas, S., Nguyen, T., Edgecomb, R. and Cheung, S.-C.: Multimedia streaming using multiple TCP connections, *ACM Trans. Multimedia Computing, Communications, and Applications* (*TOMM*), Vol.4, No.2, p.12 (2008).

[25] van der Pol, R., Bredel, M., Barczyk, A., Overeinder, B., van Adrichem, N. and Kuipers, F.: Experiences with MPTCP in an intercontinental OpenFlow network, *Proc. 29th TERENA Network Conference* (*TNC2013*) (2013).

[26] Wang, B., Wei, W., Kurose, J., Towsley, D., Pattipati, K.R., Guo, Z. and Peng, Z.: Application-layer multipath data transfer via TCP: Schemes and performance tradeoffs, *Performance Evaluation*, Vol.64, No.9, pp.965–977 (2007).

[27] Zhang, J., Gui, Y., Liu, C. and Li, X.: To improve throughput via multi-pathing and Parallel TCP on each path, *4th ChinaGrid Annual Conference, 2009, ChinaGrid'09*, pp.16–21, IEEE (2009).

[28] Zhang, M., Lai, J., Krishnamurthy, A., Peterson, L.L. and Wang, R.Y.: A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths, *USENIX Annual Technical Conference, General Track*, pp.99–112 (2004).



Che Huang received his Bachelor of Arts and Sciences and Master of Arts and Sciences degrees from Osaka Kyoiku University in 2013 and 2015, respectively. Currently, he is a Ph.D. student in the Graduate School of Information Science at Nara Institute of Science and Technology (NAIST). His current research interests include distributed systems, Software Defined Networking and related information technologies. He is a student member of IPSJ and IEEE.

Chawanat Nakasan received his B.Eng. from Kasetsart University in Thailand in 2013 and M.Eng. from Nara Institute of Science and Technology in Japan. He is currently pursuing a Ph.D. at the same university. He is currently serving as a student committee member in the Pacific Rim Applications and Grid Middleware Assembly (PRAGMA). His research focuses on multipath routing and software-defined network.

Kohei Ichikawa received his B.E., M.E., and Ph.D. from Osaka University in 2003, 2005, and 2008, respectively. He was a postdoctoral fellow at the Research Center of Socionetwork Strategies, Kansai University from 2008 to 2009. He also worked as an Assistant Professor at the Central Office for Information Infrastructure, Osaka University from 2009 to 2012. From 2012 he is working as an Associate Professor at Graduate School of Information Science, Nara Institute of Science and Technology. His current research interests include distributed systems, Software Defined Networking and related information technologies. He is a member of IEICE, IEEE, IPSJ and JSAI.

Yasuhiro Watashiba received his B.E. and M.E. degrees from Kyoto University in 2002 and 2004, respectively, and his Ph.D. degrees from Osaka University in 2015. Currently, he is working as Assistant Professor of the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). His current research interests include Resource Management on computing environment and related information technologies. He is a member of IEICE, IPSJ, IEEE and ACM.

Hajimu Iida received his B.E., M.E., and Dr. of Eng. degrees from Osaka University in 1988, 1990, and 1993, respectively. From 1991 to 1995, he worked for the Department of Information and Computer Science, Faculty of Engineering Science, Osaka University as a research associate. Since 1995 he has been with the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His current position is a Professor of the Laboratory of Software Design and Analysis. His research interests include modeling and analysis of software and development process.