Communications and Network, 2016, 8, 144-157
Published Online August 2016 in SciRes. http://www.scirp.org/journal/cn
http://dx.doi.org/10.4236/cn.2016.83015

A Low-Load QoS Routing Method for OpenFlow Networks

Shinnosuke Kimura, Kenya Sato

Graduate School of Information and Computer Science, Doshisha University, Kyoto, Japan
Email: kimshin6193414@gmail.com, ksato@mail.doshisha.ac.jp

Received 12 May 2016; accepted 26 July 2016; published 29 July 2016

Abstract

As the volume of Internet traffic continues to grow, there is a need for mechanisms that ensure communication quality that is suitable for different applications. OpenFlow implements per-flow routing control but can have adverse effects on communication quality when the number of flow entries is increased. In this study, we propose a routing method that controls traffic flows by labeling them with numbers representing the source switch, destination switch and required communication quality. The OpenFlow controller then sets up flow entries on a per-label basis rather than on a per-flow basis, allowing it to reduce the number of flow entries. In an evaluation test, we confirmed that the labeling method could prevent this reduction in communication quality.

Keywords

Routing Method, Traffic Control, OpenFlow, QoS, Labeling

1. Introduction

With the spread of applications that use the Internet, the amount of traffic carried by networks has been growing every year, and it is expected to continue growing in the future. Along with this trend, a mechanism is needed for guaranteeing communication quality so that applications can be used without problem even when there is a huge amount of traffic on the network. Internet traffic comes in various types, including text, images and video, each of which has different communication quality requirements that need to be controlled for different applications.

In recent years, Software-Defined Networking (SDN) technologies such as OpenFlow [1] have attracted attention as a way to address this issue. One of the characteristics of OpenFlow is its ability to control traffic on a per-flow basis. Once a flow has been defined, the details of packet control can be set more finely than in conventional methods that perform such control using only IP addresses. This allows traffic to be controlled more flexibly according to the needs of each application. However, this fine-grained control setting results in a greater number of flow entries, leading to a higher controller processing load and less spare capacity in the flow table. This can make it difficult to achieve further increases in traffic or guarantee the flow's communication quality.

Therefore, in this study we propose a flow-labeling method that solves the issues of increased flow entries in an OpenFlow network. The aim of this labeling method is to allow routing to be performed flexibly and with low computational load in an OpenFlow network and, moreover, to provide assurances regarding the quality of communication flows. Specifically, the same label value is applied to each flow in a set of flows having the same destination switch, source switch and required communication quality. Furthermore, routing is performed on a per-label basis in order to reduce the number of route calculations performed by the flow controller as well as the number of flow entries in the flow table.

2. Routing in OpenFlow Networks

2.1. Related Studies

A study by Egilmez *et al.* [2] proposed an OpenFlow controller design that supports end-to-end QoS (Quality of Service) guarantees. To satisfy per-application QoS requirements with regard to network traffic, they defined flows with respect to the required communication quality parameters such as lag and jitter, and assigned each flow a dynamically evaluated QoS. Dynamic QoS routing is performed by finding the constrained shortest path. For example, for a flow that requires a certain level of communication quality with regard to lag, a path would be selected that minimizes the cost subject to the requirement that the lag be kept within a certain fixed limit. The advantage of this approach is that the OpenFlow controller can use the network topology and the latest network status information such as the link costs and traffic levels to route each flow according to its required quality level and the current state of the network. When forwarding packets, the flows are identified by using multiple fields including the destination IP address, port number and ToS, resulting in the creation of a large number of flow entries.

A study by Tsuchiya *et al.* [3] proposed a method for detour routing in an MPLS network using OpenFlow. In shortest-path routing methods used in IP networks, such as RIP and OSPF, the traffic is liable to become concentrated at a specific link, so a Relay Node algorithm is used to disperse the traffic evenly to improve performance while keeping the overall network load optimally balanced. Furthermore, with a structure where MPLS (Multi-Protocol Label Switching) and OpenFlow are used together, this method combines the fast packet transfer capabilities of MPLS with the centralized network management capabilities of OpenFlow. However, it also generates a large amount of detour routing flow, and it is not assumed to guarantee the communication quality for different types of flow.

2.2. Issues

SDN and OpenFlow facilitate dynamic and flexible network control by software to deal with the increased amount of Internet traffic in recent years and the need for communication quality assurances for specific applications. However, to perform flexible network control using OpenFlow, it is necessary to provide detailed flow definitions using multiple fields, and these detailed definitions result in more flow entries. The increased number of flow entries becomes a problem in OpenFlow network routing because they adversely affect the processing load of OpenFlow controllers and the table storage capacity of OpenFlow switches.

This increased processing load on OpenFlow controllers can be attributed to the increased number of flow entries and the corresponding increase in the number of route calculations that have to be performed. Eventually, the processing load can also increase due to the route calculations and the creation of flow entries that occur when packets arrive, even when different flows are assigned to the same route. Based on the predicted increase in Internet traffic, the scalability of controllers in OpenFlow networks is liable to become a problem in the future [4]. Under such circumstances, if the processing load of an OpenFlow controller increases, its packet-in response time will increase and the communication quality of the flow will be degraded.

One reason for this stress on the table capacity of OpenFlow switches is the flow table capacity. There is a limit to the number of flow entries that can be stored in a flow table, and as the number of flow entries increases, the capacity of the flow table can be exhausted. If a required flow entry cannot be written into an OpenFlow switch, then it will not be possible to implement OpenFlow's flexible network control. Thus the communication quality of the flow will be adversely affected if the increased number of flow entries makes it impossible for the OpenFlow controller and OpenFlow switches to perform their required functions. We should therefore avoid increasing the number of flow entries in an OpenFlow network. To guarantee the communication quality of flows in an OpenFlow network, we need a method that achieves the flexible routing offered by OpenFlow while at the same time reducing the number of flow entries.

3. Proposed Method

3.1. Overview

In this study, we propose a method that resolves the issues of increased flow entries in an OpenFlow network by labeling flows and controlling the flows based on these labels. This is done by applying the same label values to sets of flows for which the destination switch, source switch and required communication quality are all the same. By performing route calculations on a per-label basis (*i.e.*, per set of flows), we can not only reduce the number of route calculations but also reduce the number of entries created in the flow table. Since this approach reduces the load on the OpenFlow controller and allows the capacity of the OpenFlow switches to be used effectively, it is able to guarantee the flow's communication quality. The aim of the proposed method is to perform flexible routing with fewer flow entries in order to implement network control with a low overhead that can satisfy the requirements of each flow.

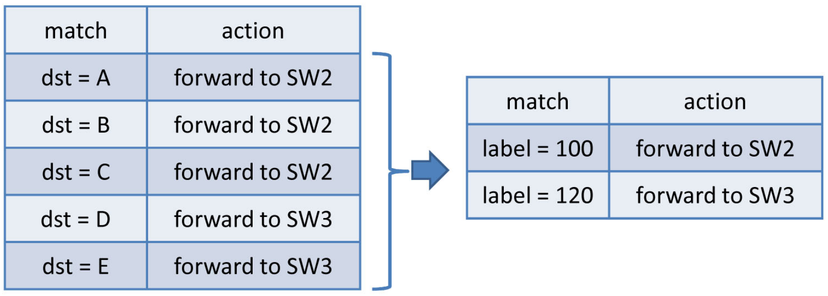
In the following, we explain how the use of labeling reduces the number of route calculations and the number of flow entries. In OpenFlow, a single flow entry is set for each destination as shown in the table on the left side of **Figure 1**. As a result, separate flow entries are set for flows with different destinations, even if they follow the same route. Therefore, labels are applied to flow entries having the same action (*i.e.*, the same route), and matching rules are created using these labels so that multiple flows can be controlled with a single flow entry as shown in the flow table on the right. Furthermore, since flows having the same label have the same path, once a path has been determined for a certain label, the OpenFlow controller can store this path information; consequently, there is no need to perform additional route calculations for other flows having the same label. This labeling method can reduce the number of route calculations that need to be performed as well as the number of flow entries.

The method proposed in this study was designed based on OpenFlow v1.0. In the following description, the controller and switch refer to an OpenFlow controller and an OpenFlow switch, respectively.

3.2. Configuration

The network configuration of the proposed method is shown in **Figure 2**. Since the proposed method is targeted at OpenFlow networks, the network is configured from a controller and switches in the same way as in the OpenFlow specification. The switches are connected to the application servers and hosts, and the application servers and hosts communicate through the network via these switches.

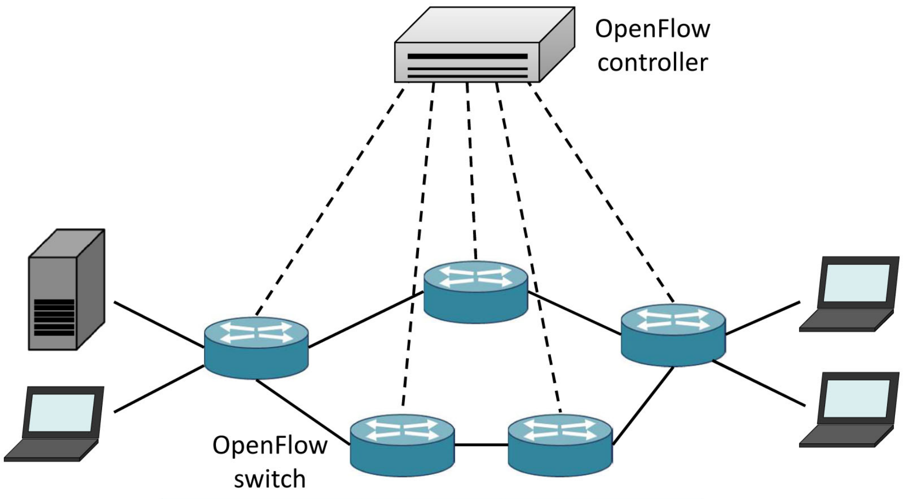
Figure 3 shows a list of controller modules used in the proposed method. In addition to the standard functions of the OpenFlow specification, the controller has a main module and four submodules that describe the overall actions of the proposed method. The main module is described in Section 3.3. The four submodules are described below.



match	action
dst = A	forward to SW2
dst = B	forward to SW2
dst = C	forward to SW2
dst = D	forward to SW3
dst = E	forward to SW3

match	action
label = 100	forward to SW2
label = 120	forward to SW3

Figure 1. An example of reducing flow entries.



match	counter	action
TCP src port = 80	100Mbyte	modify ToS 4
ToS = 4	500Mbyte	forward to SW2

Figure 2. Network configuration.

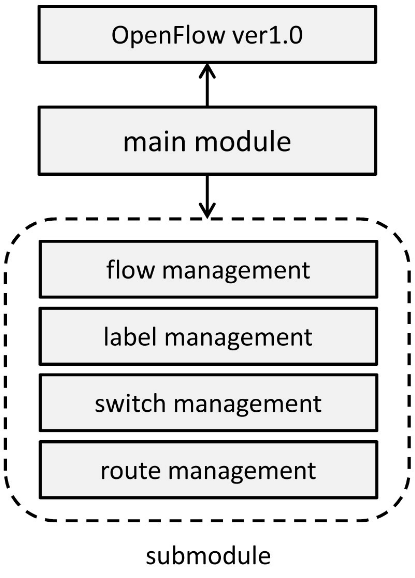


Figure 3. Module configuration.

• Flow management

Manages flows in the network. Checks the header fields of packets to determine whether they belong to QoS flows or ordinary flows. See Section 3.4 for details.

• Switch management

Manages information relating to switches in the network. Stores information such as the IP addresses, MAC addresses and physical ports of switches. See Section 3.5 for details.

• Label management

Manages the labels applied to flows. Issues new labels and stores statistical information for each label. See Section 3.6 for details.

• Route management

Manages paths between switches in the network. Also performs route calculations and stores information about the network topology and the cost values of links between switches, which is needed for these calculations. See Section 3.7 for details.

3.3. Action

The procedure whereby packets are forwarded in the proposed method is described below. An overall diagram of this procedure is shown in **Figure 4**.

1. The source host (application server) transmits a packet to an OpenFlow network switch connected to it.
2. The switch that receives this packet from the host searches the flow table for a flow entry having a rule that matches the packet?s header field. If it finds a matching flow entry, it labels the packet based on this flow entry and moves on to step (4). Otherwise, it sends a Packet-In message to the controller.
3. When the controller receives a Packet-In message, it determines the processing details of the packet included in the Packet-In message. Next, it determines a Flow-Mod message describing the processing contents and sends it to all of the switches on the route, and sends a Packet-Out message to the switch from where the Packet-In message was sent.
4. When a switch receives the Packet-Out message (having discovered the corresponding flow entry), it transmits the packet included in the Packet-Out message (based on the label value) to the next switch.
5. Thereafter, when a switch has received a packet from the previous switch, it uses the label value to forward the packet based on the flow entry in the flow table.
6. The switch that sends the packet to the destination host removes the label from the packet before sending it.

Of the packet processing performed by the controller at step (3) of the processing in **Figure 4**, the overall processing is performed by the main module, and partial processing is performed by each submodule called from the main module.

Figure 5 shows the general sequence of packet processing by the controller, and this processing is described in detail below.

• Flow judgment

Packets that include a Packet-In message are classified into either QoS flows or ordinary flows. A QoS flow is a flow that requires a certain communication quality with regard to parameters such as lag, jitter and packet loss, and an ordinary flow is a flow that has no particular quality requirements. The discrimination between QoS and ordinary flows is performed by calling the flow management module.

• Label judgment

To apply labels to packets, it is important to obtain the source switch ID, destination switch ID and flow classification. The source switch ID is the ID of the switch from where the Packet-In message was sent. As for the destination switch ID, the switch connected to the host (application server) having the destination IP address

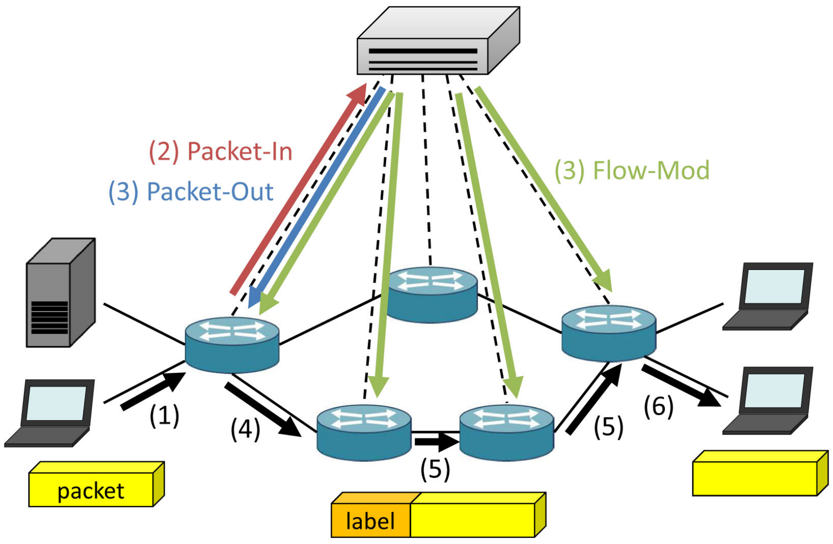


Figure 4. Processes on transfer.

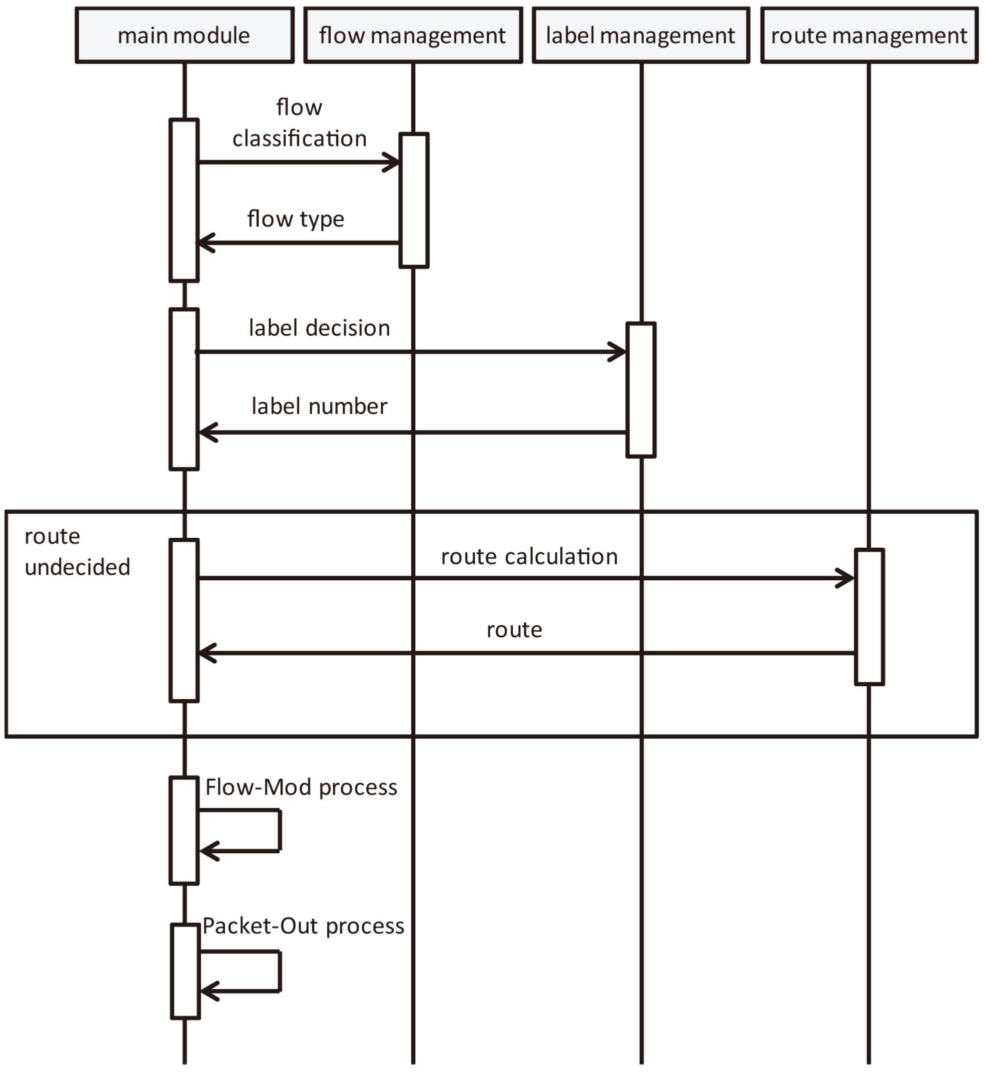


Figure 5. Processes in controller.

stored in the packet's header field checks this by calling the switch management module, and this switch's ID is set as the destination switch ID. The flow classification is determined to be either QoS flow or ordinary flow based on the flow judgment results. The destination switch ID, source switch ID and flow classification are transferred as inputs to the label management module, which chooses the label to be applied to the packet.

• Route calculation

The routes for labeled packets are determined by calling the route management module. Once a route has been decided, it is stored by the controller as the route corresponding to the label. Therefore, this processing is only performed for packets to which new labels have been applied, and if the route corresponding to a label has already been determined, then this route is chosen.

• Flow-Mod message creation and transmission

A Flow-Mod message is created and transmitted to all of the switches along the determined route. The matching rule of this Flow-Mod message and the contents set in the action differ according to the switch. **Table 1** lists the matching rules and actions set in the Flow-Mod messages.

-The source switch must transmit a Flow-Mod message describing how labeling and packet transfer should be performed based on the header field.

-The forwarding switches (*i.e.*, all switches apart from the source and destination switch) must transmit a Flow-Mod message describing how packets should be transferred based on their labels.

Table 1. Match and action in flow-mod messages.

Switch	Match	Action
Source	All	Modify IPv4 ToS bits, Forward
Destination	ToS	Forward
On the route	Ethernet destination address	Modify Ethernet destination MAC address and modify IPv4 ToS bits, Forward

-The destination switch must transmit a Flow-Mod message describing the removal of the label and the transformation of the destination MAC address.

• Packet-Out message creation and transmission

A Packet-Out message is transmitted to switches from which a Packet-In message was sent. The labeling of packets is performed by specifying the writing of label numbers in the actions of Packet-Out messages.

3.4. Flow Management

This section describes the flow discrimination of packets included in Packet-In messages, which is a function of the flow management module. Flow decisions are performed by looking up the header field of each packet in the controller's QoS table. The QoS table declares a set of rules consisting of header field combinations for which QoS control is to be performed. For example, one rule might apply to packets with a destination IP address of 192.168.0.2 and a destination TCP port number of 80. The packet header fields are each checked in turn against the rules in the QoS table, and they are assigned to QoS flows or ordinary flows depending on whether a matching rule is discovered.

In the proposed method, flows that match the rules in the QoS table are judged to be QoS flows, and the rest are judged to be ordinary flows. Therefore, flow entries for the ordinary flow may become flow entries that are applied to the QoS flow depending on the contents of the matching rules. For example, assume QoS flow packets having a destination IP address of 192.168.0.2 and a destination TCP port of 80 arrive at a switch having the flow table shown in **Table 2**. The switch searches for flow entries that match the packet?s header field, but this time the previously retrieved flow entry 1 is applied instead of flow entry 2, which would originally have been applied. This happens because the flow entries are searched in the order in which they are set. Therefore, the priority of flow entries is set according to **Table 6**. As a result, the QoS flow controller transfer performs Packet-In processing of unknown packets classified as QoS flow. By setting this flow entry in advance, it is possible to prevent the flow entries of ordinary flow from being applied to QoS flow. If this flow entry could be set as necessary in each switch when the network is activated, then it would be possible to perform flow classification using both the switches and the flow controller.

3.5. Switch Management

When transferring packets in an OpenFlow network, this is done using information such as the IP address and MAC address of the switch held by the controller. In the proposed method, this information is held by the switch management module. The switch information held by the switch management module is shown in **Table 3**. In addition to switch information, the switch management module also holds information on the host and the application servers connected to the switches. This information on the host and application servers connected to the switches is shown in **Table 4**. Information on the hosts and applications is first acquired when the packets from the host (application server) are received by a switch and when a Packet-In message is transmitted to the controller.

3.6. Label Management

This section describes the labeing of packets included in Packet-In messages as a function of the label management module. It also describes the updating of information related to labels.

Labeing involves determining the corresponding label when given a source switch, destination switch or flow classification. **Table 5** shows the label attributes used in the proposed method. The label management module stores the values of all attributes for all labels used in the network. Labelng is performed by searching the labels

Table 2. Example of priority.

Entry	Match	Counter	Action	Flow
1	IP src address = 192.168.0.0/24	100 Mbyte	Forward to SW2	General
2	IP src address = 192.168.0.2, and TCP src port = 80	10 Mbyte	Forward to SW3	QoS

Table 3. Switch information.

Attribute	Details
Datapath-id	ID of switch
IP address	IP address of switch
MAC address	MAC address of switch
Port number	All ports of switch

Table 4. Information on hosts and application servers.

Attribute	Details
MAC address	MAC address of host (application server)
Datapath-id	ID of switch connected host
Port number	Port number of switch connected host

Table 5. Label attributes.

Attribute	Details
Label number	Number of label attached flows
Source switch	ID of source switch in the flow
Destination switch	ID of destination switch in the flow
Flow type	QoS flow or general flow
Traffic	A total traffic amount of flows
Flow	The number of flows which have same label

held by the module for a label that satisfies the following requirements for the applied source switch, destination switch and flow classification.

• Applied label conditions

-Same source switch

-Same destination switch

-Same flow classification

-Traffic volume below threshold value

-Number of flows below threshold value

Threshold values for the traffic volume and the number of flows are set in advance. These threshold values are the number of flows and the total traffic volume of flows that can be given the same label.

When there is no label that satisfies these conditions, a new label is issued. For the label numbering in the proposed method, the ToS field of the IP header is used as the place where the label numbers are stored. The ToS field is one of 12 types of header fields used in the matching rules, and the types of action also include rewriting the ToS field. Since the label number is both readable and writeable, this is a suitable place for storing the label number. But although all 8 bits of the ToS field are readable according to the OpenFlow specification, only the upper 6 bits (DSCP field) can be written to. Therefore, only the upper 6 bits can be used to store the label number, and the lower two bits of the label number are set to “00” (making the label number a multiple of 4). When issuing a new label, the label number is obtained by adding 4 to the most recently issued label number.

To keep the amount of traffic and the number of flows associated with a label constantly updated, it is necessary to obtain information about the flows allocated to a label. When the network is activated, the controller periodically sends a Flow-Status message to the switches in the network. In the responding Flow-Status messages, those that apply to labeled flows are acquired, and the acquired traffic quantities are updated as the traffic quantities of these labels. When a new Packet-In message is received, the number of flows is increased by 1.

3.7. Route Management

The route management module stores the route information shown in **Table 6**. Route calculations are performed using this information. A flow?s route is determined according to the given source switch ID, destination switch ID and flow classification. Route calculations are performed using Dijkstra's shortest path algorithm, but with different parameters for QoS flows and ordinary flows.

In route calculations for QoS flows, the link costs are first redefined using the route calculation shown in Equation (1).

$$c'_{ij} = c_{ij} + t_{ij} \tag{1}$$

Here, c_{ij}, and t_{ij} are the switch IDs, cij and tij are, respectively, the cost and traffic of the link from switch i to switch j, and ω is a weighting coefficient. Since a QoS flow requires a guaranteed communication quality, it also uses dynamic parameters acquired by the controller. Using the redefined cost c'_{ij} of Equation (1), it is possible to express the cost of a route r from a source switch to a destination switch as shown in Equation (2).

$$C'_r = \sum_{(i,j)\in r} c'_{ij} \tag{2}$$

The route is decided on by finding a route $r(s,d)$ that minimizes the cost according to Dijkstra's algorithm (Equation (3)). Here, R_{sd} represents the set of all paths from switch s to switch d.

$$r(s,d) = \arg\min_r \{C'_r, r \in R_{sd}\} \tag{3}$$

Route calculations for ordinary flows are performed using only the link costs as parameters, whereby a route can be obtained by determining the minimum cost paths according to Dijkstra's algorithm in the same way as for QoS flows. Equations (4) and (5) show the formulas used for route calculations in ordinary flows.

$$C_r = \sum_{(i,j)\in r} c_{ij} \tag{4}$$

$$r(s,d) = \arg\min_r \{C_r, r \in R_{sd}\} \tag{5}$$

Moreover, when the network is running, and during activation, route information such as the network topology and links is acquired. When the network is running and the controller has recognized the connections between the switches, it transmits a Features request message to the switches. On receiving this message, a switch sends a Features response message back to the controller. The Features response message stores information on the switch's physical port, and on receiving this information, the controller is able to ascertain what sort of physical ports the switch has. When a switch's physical port can be ascertained, the controller sends an LLDP frame to the switch with a Packet-Out message. On receiving the LLDP frame, the switch broad-

Table 6. Route information.

Attribute	Details
Source switch	ID of link source switch
Destination switch	ID of link destination switch
Port number	Port number of link source destination switch
Cost	Cost of link (distance, delay)
Traffic	Traffic amount

casts to all of the switch's physical ports. When a switch receives an LLDP frame broadcast in this way, it sends an LLDP frame to the controller with a Packet-In message. On receiving the LLDP frame, the controller can know the correspondence between the link's source and destination and the source's physical port number. When the network is started up, the controller transmits a Port-Status message to the switches at fixed intervals, and depending on the responses from the switches, it is able to obtain the quantities of traffic transmitted by each port of the switches. The values obtained here are taken to be the link traffic quantities.

4. Evaluation

4.1. Evaluation Environment

In this study, we performed an evaluation with a simulation using the virtual network functions of Trema [5] and Linux. Trema is a framework for the development of OpenFlow controllers, which also has functions for the construction of virtual networks with virtual hosts and switches [6]. We used Ruby as the development language. The parameters of the evaluation environment are listed in **Table 7**.

4.2. Evaluation Scenario

In the evaluation, we verified the validity of using labels for flow control in an OpenFlow network. We prepared a network using multiple virtual servers and virtual hosts, and we transmitted packets from the virtual servers to the virtual hosts to evaluate the controller?s activity and the flow?s communication quality. We prepared the evaluation scenario shown in **Table 8** and performed the evaluation five times while varying the number of virtual hosts between 10 and 100. **Figure 6** shows the network topology of the evaluation scenario.

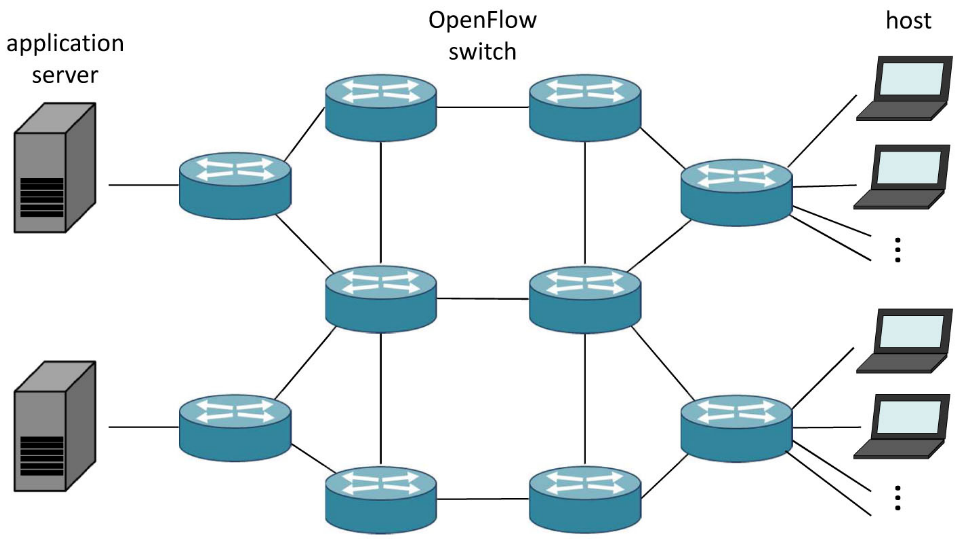


Figure 6. Network topology.

Table 7. Evaluation environment.

Attribute	Details
OS	Ubuntu 14.04 LTS
CPU	Intel Core i3 CPU 550@ 3.20 GHz × 4
Memory	3.7GiB
OpenFlow controller	Trema Version 0.4.7
OpenFlow switch	Open vSwitch 2.0.2
Benchmark tool	Iperf3 Version 3.0.7

Table 8. Evaluation scenario.

Attribute	Details
OpenFlow controller	1
OpenFlow switches	10
Application servers	2
Hosts	10 - 100
Time	60 s
IP protocol	TCP
Transmitted data	100 kbps

In this topology, there are multiple hosts connected to a single switch, and the topology can be verified by the effects of flow integration. We used 6 bits of the ToS field to store the label numbers in this study so that only 64 different label numbers would be created; therefore, we set the scale of the network so that larger numbers of routes would not occur. For comparison, we also performed packet transfer using conventional header fields defined according to the OpenFlow specification without the labeling method proposed here.

4.3. Evaluation Results

As the evaluation results, **Figures 7-9** show the number of times the controller performs route calculations, the number of flow entries created by the controller, and the throughput, respectively. The evaluation results are average values obtained by performing the evaluation five times with a set number of hosts.

5. Discussion

5.1. Discussion of Evaluation Results

The evaluation results show that, compared with the conventional method, the proposed method reduces both the number of route calculations performed by the controller and the number of flow entries created. This is because the labeling performed in the proposed method facilitates the storage of route calculation results and the integration of multiple flows. On the other hand, there was no large difference in throughput. This is because although the proposed method and the conventional method differed with regard to the controller processing load and the number of flow entries held by the switches, these differences affected the network's packet transfer performance more than they did the communication quality. In this evaluation, within the range of the network transfer performance, we show that there was no loss of communication quality when performing labeling by the proposed method.

Factors that can be used to determine the communication quality of an OpenFlow network include the controller processing load, the performance of the control network (secure channel), and the bandwidth/ transmission rate of the transport network. The method proposed in this study improves the communication quality when the controller processing load becomes a bottleneck. It is important to ascertain the performance of the controller and network, and in OpenFlow this can be done by exchanging messages between the controller and switches to obtain related information that can be used to control the network. The only new requirement for labeling in the proposed method is the availability of the ToS field in the IP header. The other fields can be used without any change from the OpenFlow specification. It is therefore possible to apply the proposed method to a particular part of a network where the controller has an increased processing load, instead of being used throughout the entire OpenFlow network. The other parts of the network can be operated as a conventional OpenFlow network where flows are defined using IP addresses, MAC addresses, port numbers and the like.

5.2. Comparison with Related Technologies and Related Studies

MPLS is a packet transfer technique that uses labels. In MPLS, labels are applied as identifiers between the data link and network layers of the OSI reference model, and packets are transferred based on the label values. MPLS

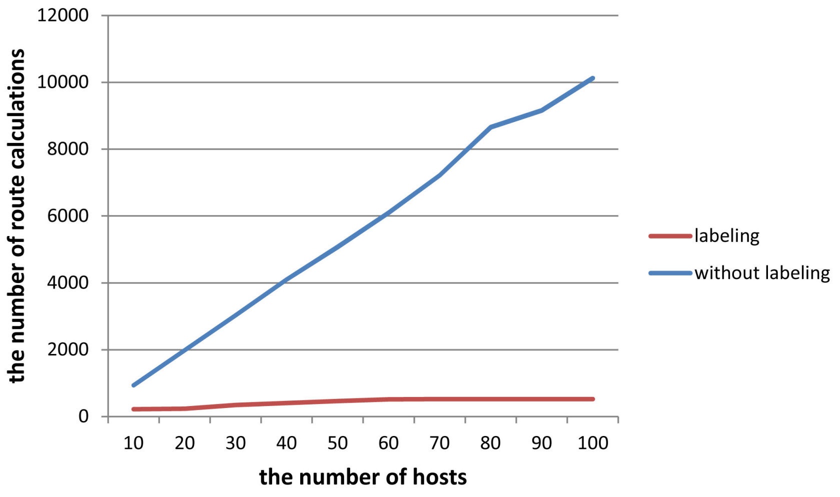


Figure 7. Number of route calculations.

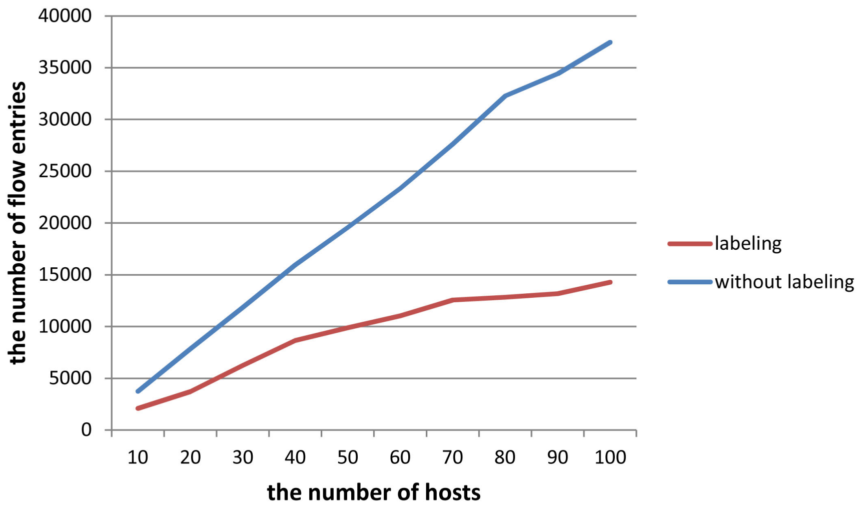


Figure 8. Number of flow entries.

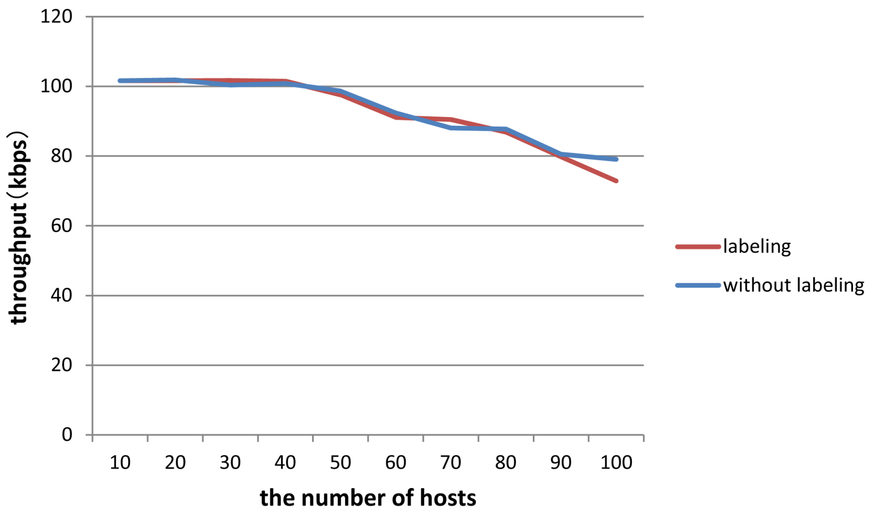


Figure 9. Throughput.

is not a technique that can be used across an entire IP network, but a mechanism whereby labels are added on entering the MPLS network and removed when they leave it. The proposed method is constrained to use fields that can be used as labels because it is based on the OpenFlow specification, so MPLS is more flexible in this respect. Also, MPLS replaces labels whenever a packet is transferred to an LSR (label switch router), but in the proposed method labels are applied in flow units, so label replacement does not occur when packets are transferred. The replacement of labels incurs delays due to the processing overheads, but makes it possible to integrate flows having different source addresses. The proposed method is targeted at OpenFlow networks, so the distribution of label information can be performed collectively by the controller.

In a study by Egilmez *et al.* [7], routing was performed in an OpenFlow network with a focus on video streaming. They define three flow types, and for flows that provide a QoS guarantee, the routes are determined by solving the shortest-path problem with the additional constraints of minimizing jitter and keeping the delays within a set limit. In this method, since the flows are classified into three types, the route calculations involve a large amount of computation that increases the burden on the OpenFlow controller and also results in more flow entries being set. In a study by Kim *et al.* [8], QoS control is performed by using flow integration and priority queues in an OpenFlow network. Studies that perform QoS control by integrating OpenFlow with MPLS include those of Khan *et al.* [9] and Lin *et al.* [10]. The study by Khan *et al.* uses a configuration with an OpenFlow switch at the entrance to an MPLS network as well as OpenFlow's programmable control features to perform the actual transfer of packets according to the MPLS specification. The integration of both MPLS and IP networks with OpenFlow is problematic, but a study by Kitada *et al.* [11] investigated a method whereby communication could be continued by having one protocol be interpreted by the other, even when it has become necessary to alter the path of a flow due to a problem such as the disconnection of a link.

6. Conclusions

The traffic carried by networks has been increasing every year, and network operators are looking for ways to control networks according to the type of traffic they are carrying. In recent years, the OpenFlow standard has risen in popularity, allowing networks to perform flexible control on a per-application basis. On the other hand, since many flow entries are needed to perform flexible control, the increased number of flow entries can cause problems with regard to the processing load on the OpenFlow controller and the capacity of the flow tables in OpenFlow switches.

Therefore, in this study we proposed a method where labeling is used to perform routing in an OpenFlow network. The use of labeling can reduce the number of route calculations performed in the OpenFlow controller and the number of flow entries in the flow table. In an OpenFlow network, since flow entries are created for each destination represented by a matching rule, separate flow entries are set up for multiple flows traveling along the same routes. The proposed method reduces the number of flow entries by using labeling so that flow entries can be set for individual routes through which flows pass instead of for individual destinations. In addition, since the routes are determined on a per-label basis, it is possible to use the flow entries of other flows whose route calculation has already been assigned to the same label, thereby reducing the number of times route calculations need to be performed. Furthermore, since the flow definitions are performed using diverse information in packet header fields in the same way as in a conventional OpenFlow network, it is also possible to achieve the flexible control that characterizes OpenFlow networks. In the proposed method, flows are classified into QoS flows and ordinary flows, and route calculations are performed differently depending on the class of flow.

We evaluated the proposed method by performing a simulation in an emulator using the virtual network functions of Trema and Linux to compare it with the conventional method. Compared with the conventional method, we confirmed that the proposed method could reduce the number of route calculations and the number of flow entries without any major loss of throughput. In this evaluation environment, we reached the performance limits of the virtual network before we saw any effect on communication quality due to differences between the proposed method and the conventional method, and within the range of packet transfer functions in the virtual network, the use of the proposed method was found to have no effect on communication quality. This demonstrates the validity of the proposed method.

Acknowledgements

This research was supported in part by JSPS KAKENHI Grant Number JP26540038 and JP16H02814.

References

[1] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J. (2008) OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, **38**, 69-74. http://dx.doi.org/10.1145/1355734.1355746

[2] Egilmez, H.E., Dane, S., Bagci, K. and Tekalp, A. (2012) Open QoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks. *Signal & Information Processing Association Annual Summit and Conference* (*APSIPA ASC*), Asia-Pacific, 1-8.

[3] Tsuchiya, T. and Kawarasaki, M. (2015) Relay Node Based Routing Control Using Mpls-OpenFlow Architecture. *IEICE Technical Report*, **114**, 25-30.

[4] Nunes, B.A.A., Mendonca, M., Nguyen, X.-N., Obraczka, K. and Turletti, T. (2014) A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *Communications Surveys & Tutorials*, **16**, 1617-1634. http://dx.doi.org/10.1109/SURV.2014.012214.00180

[5] Trema: Full-Stack OpenFlow Framwork in Ruby and C. https://trema.github.io/trema

[6] Fernandez, M.P. (2013) Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. *IEEE* 27*th International Conference on Advanced Information Networking and Applications* (*AINA*), Barcelona, 25-28 March 2013, 1009-1016.

[7] Egilmez, H.E., Civanlar, S. and Tekalp, A. (2013) An Optimization Framework for QoS-Enable Adaptive Video Streaming over Openflow Networks. *IEEE Transactions on Multimedia*, **15**, 710-715. http://dx.doi.org/10.1109/TMM.2012.2232645

[8] Kim, W., Sharma, P., Lee, J., Banerjee, S., Tourrilhes, J., Lee, S.-J. and Yalagandula, P. (2010) Automated and Scalable QoS Control for Network Convergence. 2010 Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN 10).

[9] Khan, A., Kiess, W., Perez-Caparros, D. and Triay, J. (2013) Quality-of-Service (QoS) for Virtual Networks in OpenFlow MPLS Transport Networks. *IEEE* 2*nd International Conference on Cloud Networking* (*CloudNet*), San Francisco, 11-13 November 2013, 10-17. http://dx.doi.org/10.1109/cloudnet.2013.6710552

[10] Lin, H., Sun, L., Fan, Y. and Guo, S. (2012) Embedded OpenFlow MPLS-Based Advance OpenFlow Inter-Domain Communication Proposal. 8*th International Conference on Networking and Mobile Computing* (*WiCOM*), Shanghai, 21-23 September 2012, 1-4.

[11] Kitada, H., Kojima, H., Takaya, N. and Matsubayashi, Y. (2013) A Study on Routing Method for ip OpenFlow Hybrid Network. Tech. Rep. 463, IEICE Technical Report.