Traffic Control System Based on SNS Information in a Deeply Programmable Network

Haruka Yanagida
Ochanomizu University
Bunkyo, Tokyo, Japan
haruka@ogl.is.ocha.ac.jp

Akihiro Nakao
University of Tokyo
Bunkyo, Tokyo, Japan
nakao@iii.u-tokyo.ac.jp

Shu Yamamoto
University of Tokyo
Bunkyo, Tokyo, Japan
shu@iii.u-tokyo.ac.jp

Saneyasu Yamaguchi
Kogakuin Univirsity
Shinjuku, Tokyo, Japan
sane@cc.kogakuin.ac.jp

Masato Oguchi
Ochanomizu University
Bunkyo, Tokyo, Japan
oguchi@is.ocha.ac.jp

***Abstract*—When the Great East Japan Earthquake occurred in 2011, telephone and Internet could not be used in many cases. To achieve network availability in disaster situations, we propose the traffic control system based on SNS information. In order to utilize the data of packet payload obtained from SNS, deeply programmable network (DPN), which is extending SDN as implementing programmable data plane and API (Southbound Interface), is needed. In this paper, we use the FLARE switch, which performs not only the function of OpenFlow but also the traffic classification based on the application, using data plane programmability and the content-based control of the packet payload. By using our proposed system, more flexible network control is achieved.**

***Index Terms*—SDN; OpenFlow; DPN; FLARE; SNS;**

I. Introduction

In recent years, Internet traffic has increased because of the spread of high-performance mobile devices, such as smartphones and tablets, and the development of cloud computing. In particular, mobile video traffic has been increasing rapidly, and the amount will increase 13-fold between 2014 and 2019. The increased traffic may increase the risk of occurrence of Internet traffic congestion [1].

Network congestion can be a serious problem, especially in an emergency, such as an earthquake. Such large-scale disasters often cause congestion and network failures because base stations and network facilities are damaged and many users are trying to access the network at the same time. In the case of an emergency, it is important that telephone and Internet be available. Usually, high availability (HA) networks are implemented with redundancy or scalability of the network. However, when the Great East Japan Earthquake occurred in 2011, network disconnection occurred in some areas because the reconfiguration of the switching required manual operation, and it was difficult to grasp all network conditions immediately [2]. This incident made people realize the importance of a stable network control system that can operate even in large-scale disasters. Therefore, the necessity of a system that can grasp all network conditions immediately and is entirely automatic became more apparent in Japan.

Several researchers have applied SDN and OpenFlow [3] technology to wide area networks to centralize the management and control of network devices, such as routers and switches, using a software controller. However, the following two challenges must be considered. First is the difficulty of detecting network failure by only monitoring using sensors inside of a network when the target area is wider and the network is complicated. Second is the limit of programmability. The advance of technology has enabled control plane(C-plane) to be programmable, whereas the data plane (D-plane) programmability was not considered. C-plabe programmability expanded the flexibility of the network control. C-plane programmability, however, has the limitation for the advanced packet manipulation such as the DPI control at the switching or routing devices. Recently, the data plane (D-plane) programmablility is being investigated as well [4]. Therefore, using D-plane programmabiliy, we considered more flexible control methods such as application aware flow control which is useful for the priority control for the shortage of the available network resources in the case of emergency.

Against this background, we propose the traffic control system using Social Networking Service (SNS) information in a Deeply Programmable Network (DPN) to address these challenges. We explain our system based on two major points. First, we use SNS information to detect network failure in addition to the monitoring of network devices. We previously detected failure with high accuracy using Twitter [5] in real-time [6]. We perform traffic control based on this method because it can specify the area of network failure and grasp all network conditions immediately. This process is one solution for the first challenge. In research on the [6], Maru explained the reasons why the collective intelligence of Twitter is suitable as a means of information detection complementary to conventional observation. The second point is the DPN. DPN is the concept of a software-defined D-plane, meaning the full programmability of a network. DPN removes the limit of network programmability and is the solution for the second challenge. This D-plane programmability is needed because we focused on advanced and flexible network control, e.g., application specific traffic control utilizing SNS information. In disaster situations, it is assumed that prioritizing important traffic, such as mail, phone, or SNS, rather than mobile video traffic which has currently increased explosively, is effective. In order to achieve this type of application specific traffic control, data-plane functionality must be extended from the current SDN model where D-plane elements have limited pattern match capabilities and too few actions.

In this paper, we use the FLARE switch [7]-[8] to achieve the DPN environment and implement/ evaluate both a Twitter network failure detection system and a traffic classification system. Figure 1 shows network environment including the proposed system. As shown in this figure, SNS information is collected and analyzed. Based on those, the Network Controller makes a decision for routing and bandwidths control for each slices where are applied each applications, so that it sends a direction to switches on the network.

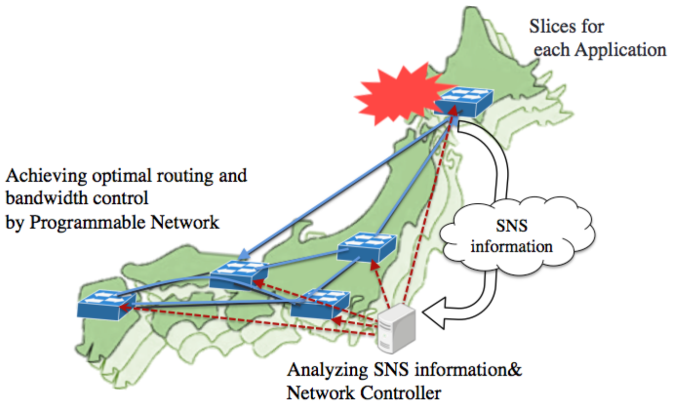

Slices for each Application
Achieving optimal routing and bandwidth control by Programmable Network
SNS information
Analyzing SNS information& Network Controller

Fig. 1. Network environment achieved by the proposed system

Our contributions in this paper are as follows.

i Integration of SNS-based Failure Detection [6] into Network Control. We prove that it is effective to use the network failure information extracted by the real-time analysis of SNS in addition to the monitoring of the network devices.

ii Advanced network software control, such as application classification and QoS control. By applying DPN, we can classify the application and/or device specific traffic into slices and apply fine-grained quality of services (QoS) assessment.

The rest of the paper is organized as follows. Section II introduces related research studies, and Section III explains the OpenFlow and FLARE mechanisms and compares them. Section IV gives an overview of our proposed system. Section V describes the experimental environment briefly, and Section VI and VII discuss various experiments based on our proposed system. Finally, in Section VIII, we conclude with a brief summary and mention the future direction of our research.

II. Related Work

There are several methods that control networks automatically in disasters using SDN and OpenFlow software controller [9][10][11][12][13][14][15]. NTT docomo et al. [9] achieved QoS by using OpenFlow, squeezing the low priority traffic. Specifically, class-based-QoS by DiffServ (Differentiated Services) marks the QoS class to identify between the communication flow of the receiving packets and class. Using this system, they calculated the priority of the packet. However, this priority control cannot be used under situations where one user uses a number of communication services. This means it can not classify the type of applications exactly, therefore, this work differs from ours. We identify the type of applications under any circumstances and perform optimum control for each of them. Ozawa et al. [10][11] proposed management of the table of mapping IP address and GPS information on DB and coordinating DB and the OpenFlow controller to prioritize the traffic in disaster areas. They embedded a disaster ID into the ToS field in the IP header to separate the traffic and achieve QoS. Moreover, in terms of using external information, such as Earthquake Early Warning, this research resembles our own work. However, our research has the unique feature of using the collective intelligence of SNS for network control, which is based on the contents of application. In regards to [12][13][14], they applied OpenFlow to ad hoc wireless networks and, based on scores of priority weight that users determine in advance, proposed QoS. Edo et al. [15], by modeling risks of disaster with a numerical expression, had an approximately 1.66 times performance improvement compared with the controls using the shortest path only. However, these studies are based on network internal information, such as traffic monitoring data, which is different from our method of using external information. In our work, the entire process from obtaining SNS information to network control is done automatically and autonomically.

Furthermore, the related works introduced above are limited to the SDN concept, which is just the centralized management of network devices. This means they have no idea how to operate unique control inside of a network devices by using programmability of the D-plane. On the point of using freely rewritable software switches, the methods differ markedly. These studies could not achieve a fine-grained level of control based on the communication content. Applying the DPN concept can enable control based on the application contents of the packet.

III. SDN and DPN

A. *Control Using SDN/OpenFlow*

OpenFlow [3] is the one of the algorithm of SDN, enabling central control of a network. This technology has already been commercialized, mainly in data centers providing cloud service and enterprise LAN.

OpenFlow separates the C-plane from the D-plane. C-plane calculates and determines the data routing path, and the D-plane forwards data following the C-plane instructions. These two functions are integrated into a conventional hardware switch. In contrast, in OpenFlow C-plane is implemented as software on the server outside of the switch, ordering the D-plane inside of switch to transmit data. OpenFlow is the standard interface to connect these two parts.

B. *Control Using DPN/FLARE*

According to Section III-A, the D-plane executes only forwarding packets. The D-plane is not programmable and is fixed in SDN and OpenFlow. DPN is the concept of a software-defined D-plane to remove the limit of network programmability and make the network fully programmable.

To achieve this DPN, FLARE [7] has developed at the Nakao laboratory at University of Tokyo. With this technology, by programming the D-plane and adding the required functions, anyone can make original network devices.

In FLARE, the D-plane is implemented with a Click module router [16], which is software module router. Click is a language that creates network devices using software, such as switches and routers. Some basic functions, like ”Receiving frame”, ”Transmitting frame”, and ”Referring routing table”, which are preliminarily prepared as modules, are the features of Click. By combining modules, you can script the operation of a network device. In addition, you can make your own modules to easily create unique devices. In the Nakao laboratory, they implemented OpenFlow 1.3 software switches by Click module, which enabled the use of OpenFlow on FLARE.

Figure 2 shows the mechanism of FLARE. The software switch is implemented by using Click elements in the container for which we use the word of Sliver, and the virtual network called Slice is made by combining Slivers.

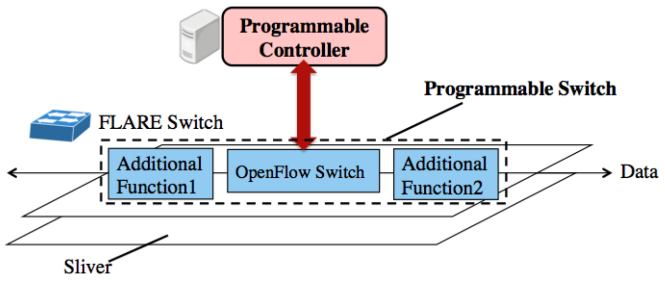


Fig. 2. FLARE method

C. Comparing SDN with DPN

DPN/FLARE enables deeper programmability of the network than SDN/OpenFlow.General switches, including OpenFlow, transmit by examining from layer 1 to 4. By contrast, the FLARE switch can examine all layers, including a part of the data. In other words, OpenFlow only handles up to the Transport layer, whereas FLARE processes the whole layer in the Application layer. For this reason, FLARE achieves more flexible network control, like application-based control, by distinguishing the application. Hence, FLARE is a suitable platform because the purpose of our study is to achieve advanced and flexible network control based on SNS information. FLARE utilizes the data of the application layer and data payload.

D. Application Classification on FLARE

There are several ways of classifying applications on FLARE. We introduce the one described in [17][18]. In this way, we attach meta information, such as the application name or device type, to the end of the packets as a trailer (a header is not recommended) among the users of the infrastructure. When these packets reach FLARE, by accessing the metadata, we can identify the application. Therefore, we achieve network control based on application type.

IV. Overview Of The Proposed Method

In this paper, we propose an advanced network control system that is based on a Twitter Failure Detection system [6] and optimizes traffic from every application. An overview of our proposed system is given in Figure 3. (1)-(4) is implemented by python on a controller, (5) is implemented by click on switch, and all executions are automated. The process flow of the proposed method is as follows.

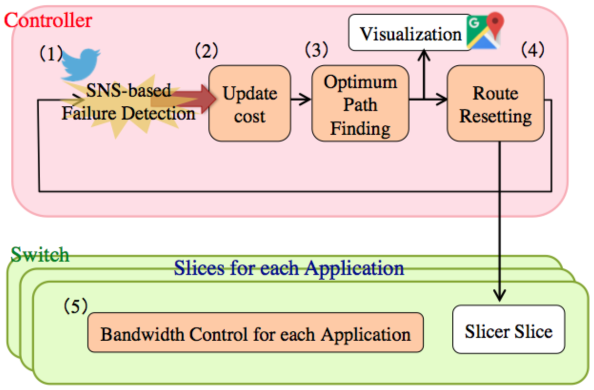


Fig. 3. Proposed advanced network control system

(1) Receive network failure information from analyzing SNS
By the system discussed in [6], we analyze Twitter in real-time and detect network failure with high accuracy. From the extracted information, we receive the number of tweets and the area of network failure.
(2) Update the costs of links
Update the cost of links following the condition. For example, the default costs of all links are 1. If there are more than 20 tweets including the mapped area name in an extracted tweet, update the cost of links by + 1. Updating is performed at 60-seconds intervals.
(3) Optimal route search
Search the optimal route by Dijkstra’s algorithm based on the updated cost. The minimum cost of a route from the start to the goal is set as the optimal route. (A lower costs indicates that the available bandwidth is larger.)
(4) Route resetting
The optimal route is reset by applying flow entry to the switch through REST-API.
(5) Application QoS control
We classify applications and assign one application to one slice [17][18]. For each slice, set D-plane as optimal bandwidth programmed by Click module router.

These all processes are executed automatically and autonomically. In the following sections, we will perform the experiment using FLARE switches in order to verify the basic autonomous switching capability by using the proposed method mentioned above.

V. Experiment Environment

A. FLARE Experiment Environment

We conduct experiments in the network environment shown in Figure 4. FLARE Switch1 and Switch2 have 8 NICs of 1G and 2 NICs of 10G for each (1G indicates 1Gbps connection, 10G indicates 10Gbps connection). FLARE Switch3 and Switch4 have 4 NICs of 10G for each. These four switches are combined as a mesh topology with mixing of 1G and 10G. FLARE Central, shown in Figure 4, is the management server of FLARE, where we can create sliver and slice through GUI. The controller shown in the red part of the proposed system in Figure 3 is implemented on this FLARE Central server. Through this controller, the four FLARE switches are controlled, and various control models are verified. Table I shows the specifications of the machines composing the experiment environment.

TABLE I
Specifications of Machines

FLARE switch1 ～ FLARE switch4	CPU	Core i7-3612QE Mobile 2.1GHz
	Memory	8GB
	OS	CentOS 6.4
h1～h4	CPU	Core i5-4210 M 2.6GHz
	Memory	8GB
	HDD	SATA 500GB 5400RPM
	OS	Ubuntu14.04
h5, h6	CPU	Xeon E3-1241 v3 3.5GHz
	Memory	8GB
	HDD	SATA 1TB 7200RPM
	OS	Ubuntu14.04

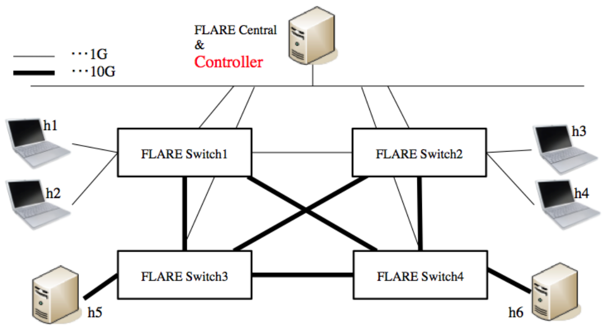


Fig. 4. Network Physical Diagram

B. Throughput Measuring

In this section, to test the behavior and to determine the attribution of the FLARE environment explained in Section V-A, the throughput is measured by iperf of each route. The outcome regarding throughput is given in Table II. The route with mixed 1G and 10G had decreased performance compared with the 1G only or 10G only routes. For this reason, the combination of 1G and 10G is a bottleneck. NOTE: These throughputs are not related to the cost introduced in the next section.

TABLE II
Throughput of Each Route

	Route	Throughput
1G10G mixed	h1-s1-s3-h5	500(Mbits/sec)
	h1-s1-s2-s3-h5	590(Mbits/sec)
	h1-s1-s4-s3-h5	500(Mbits/sec)
	h1-s1-s2-s4-s3-h5	500(Mbits/sec)
1G only	h1-s1-h2	930(Mbits/sec)
	h1-s1-s2-h3	930(Mbits/sec)
10G only	h5-s3-h6	2.1(Gbits/sec)
	h5-s3-s4-h6	1.7(Gbits/sec)

VI. Experiments and Evaluation

In this section, three experiments are showed as follows.

- Proposed system verification experiment
- Throughput evaluation experiment
- RTT evaluation experiment

First, our proposed system is described in detail by a verification experiment. After that, to evaluate this system, we measure the throughput. As a last experiment, RTT is measured to determine the time for the switching route.

A. Verifying the Proposed System Experiment

In this study, the network control system by FLARE environment shown in Figure 4 is implemented on a wide area network, as shown in Figure 7, which is emulated on our experiment environment. In this section, we provide an example of the behavior of this system, including visualization of the experimental outcome.

1) Verifying the Proposed System: We verify our proposed system of (1)-(4) shown in Figure 3. In this experiment, (5) is operating OpenFlow1.3 by the Ofswitch module of Click. Accordingly, the SDN level experiment is executed on FLARE, which has the capability of DPN. In this experiment, we use actual tweets from 14:00 to 15:00 on 11 March 2011 when the Great East Japan Earthquake occurred. We map from FLARE Switch1 to 4 to "Iwate", "Kyoto", "Tokyo", "Fukuoka", respectively. For example, the start point of communication is set to host1 near Iwate, and the goal point is set to host5 near Tokyo. Through this experiment, we verified the proposed system can switch the routes which go around the disaster area in a disaster situation based on Twitter information. Specific operations are as follows.

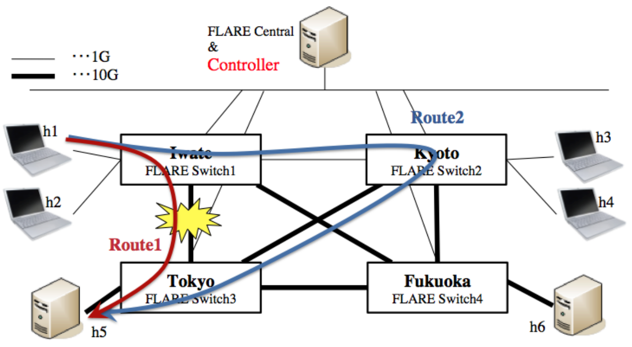
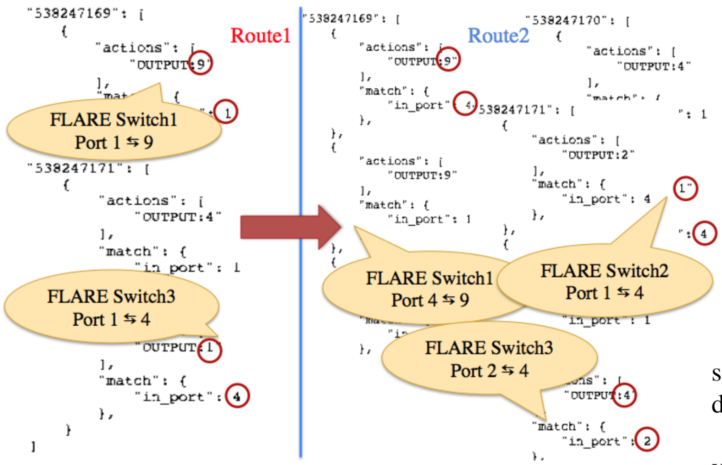


Fig. 5. System Verifying Experiment

Firstly, Route1 shown in Figure 5 (h1-s1-s3-h5) is used because the default cost of all links is 1, so the minimum cost of Route1 is 1. Then, 120 seconds (second update) later, the link cost between Iwate and Tokyo is updated to 2 because the Twitter analyzing system detects more than 20 tweets related to network failure in Iwate and Tokyo around this time. The tweets in Iwate and Tokyo continue increasing, so the cost of Route1 is incremented. 180 seconds (third update) later, the cost between Iwate and Tokyo become 3, so Route2 (h1-s1-s2-s3-h5) is selected because Route2 has a cost 2, which is the minimum. As a result, when Route2 is chosen by Dijkstra's algorithm, the route is switched by REST-API, which sets the route to Route2.

2) Checking Port Number for Sure: From the experiment in SectionVI-A1, the switching was successful. We prove the success using Figure 6 to capture the switches statement before and after the disaster. In our system, we switch the route by changing ports through REST-API. Figure 6 shows flow tables of the switches from left of the blue line, which represents before the disaster, to the right of the blue line, which represents after the disaster. The port numbers are changed before and after the disaster. Therefore, switching from Route1 to Route2 is achieved.



```
"538247169": [
    {
        "actions": [
            "OUTPUT:9"
        ],
        "mat         {
                       ": 1

"538247171": [
    {
        "actions": [
            "OUTPUT:4"
        ],
        "match": {
            "in_port": 1

            "OUTPUT:1"
        ],
        "match": {
            "in_port": 4
        },
    }
]
```

```
"538247169": [
    {
        "actions": [
            "OUTPUT:9"
        ],
        "match": {
            "in_port": 4
        },
    },
    {
        "actions": [
            "OUTPUT:9"
        ],
        "match": {
            "in_port": 1
        },
    },
    {
```

```
"538247170": [
    {
        "actions": [
            "OUTPUT:4"
        ],
        "match": {
                       ": 1
"538247171": [
    {
        "actions": [
            "OUTPUT:2"
        ],
        "match": {
            "in_port": 4
        },
    },
    {
                       ": 4
            "in_port": 1
```

```
            "OUTPUT:1"
        ],
        "match": {
            "in_port": 4
        },
    },
    {
                       ": 4
            "in_port": 1
        ons": [
            "OUTPUT:4"
        ],
        "match": {
            "in_port": 2
        },
```

Fig. 6. Changing the Port Number before and after the disaster

3) Visualization Output of Switching Route: Figure 7 shows a visualization of the output of the switching route shown in Figure 3. We visualize the routing on Google Maps in real-time based on the optimal route calculated in (3). Using this system, you can see the switching from Route1 to Route2 on the GUI.

B. Throughput Evaluation Experiment

Throughputs are measured using iperf to evaluate the proposed system. In Route2, throughputs of the proposed system are 588 Mbits/sec, while the system under normal conditions are 590 Mbits/sec. Because of almost no difference between them, we confirm that the proposed system achieves almost full performance of Hardware without overhead.

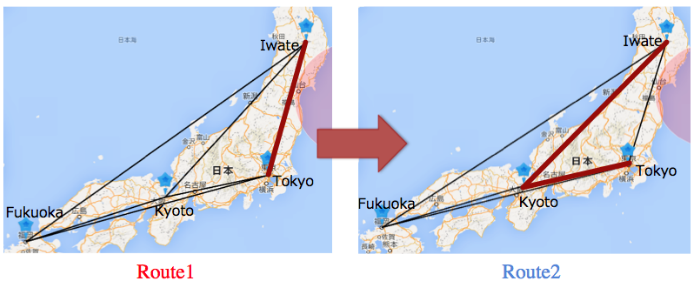


Fig. 7. Visualization routing web page

C. RTT Evaluation Experiment

To confirm the time of switching the routes, RTT are measured using a program sending Ping every one second. The outcome is shown in Figure 8. Approximately 20-30 ms RTT to switch routing can be seen because the packets do not take a new right path while route is in the middle of switching.

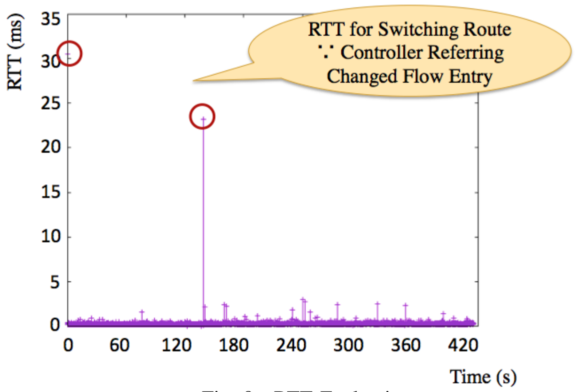


Fig. 8. RTT Evaluation

VII. Discussion

In the previous section, we confirmed the basic autonomous switching capability using SNS information. In this section, we discuss about application QoS control.

To detect the application of the traffic flow, we had developed the mechanism to add the application indicted trailer bytes to the packet in the user devices. The FLARE switch can parse the trailer and control the traffic using D-plane programmability. Even for the encrypted user traffic, FLARE switch can detect the application because the trailer bytes are not encrypted [18].

After classifying applications into slices, we set optimal bandwidth for each slices. To set the bandwidth, the click program combines the BandwidthRatedSplitter module specifying the bandwidth and the Ofswitch module that implements OpenFlow1.3. Figure 9 illustrates the application based traffic classification and QoS control. For example, we can prioritize the less bandwidth but informative traffic such as mail, SNS, and voice applications. On the other hand, we can reduce the video traffic requiring the large bandwidth. This control is of great importance for the shortage of network resources in the occurrence of multiple network failures due to the big disaster.

Thus, in disaster situation, traffic control for each application is realized against the control for all traffic in the conventional method.

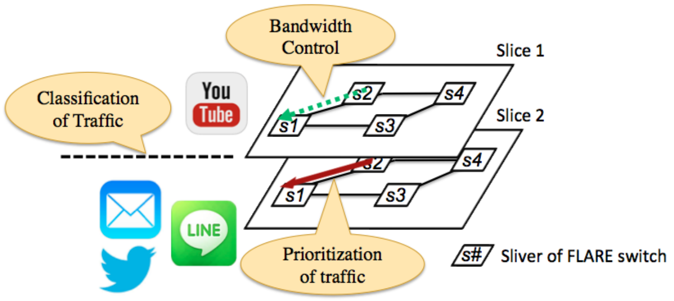


Fig. 9. Illustration of application based traffic control by using deeply programmable nodes

VIII. Conclusion

In this paper, we propose an advanced, fully programmable network control system that is based on Twitter Failure Detection, which optimizes traffic for each application automatically and autonomically. Our contributions in this paper are as follows.

As a first contribution, we built a system for switching routes to avoid a disaster area based on Twitter information. From the experiments, we verified and evaluated this system, and two things below are observed in the environment implemented with four FLARE switches. First, the proposed system achieves almost full performance of Hardware without overhead. Second, switching requires approximately 20-30 ms RTT. Hence, we confirm that our system can be operated with sufficient performance on a wide area network test bed (JGN-X[19]) implemented with seven FLARE switches.

The second contribution is to show the capability of the application based QoS system by using D-plane programmability although the experimental verification is left as the future study.

As a next step, we plan to extract more detailed situations of users from Twitter and achieve unique control reflecting that information. With this work, more advanced and flexible fine-grained network control shall be achieved.

Acknowledgment

This work is partially supported by the Strategic Information and Communications R&D Promotion Program (SCOPE).

References

[1] Cisco Visual Networking Index(VNI), "Global Mobile Data Traffic Forecast Update, 2014-2019", White Paper, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf, Feb. 2015.

[2] NTT DOCOMO, "Improvement of Credibility for Op eration System in the Case of Large Disaster", https://www.nttdocomo.co.jp/binary/pdf/corporate/technology/rd/technical_journal/bn/vol20_4/vol20_4_026jp.pdf, Technical Journal Vol. 20 No. 4, 2013.

[3] N.McKeown, T.Anderson, H.Balakrishnan, G.Parulkar, L.Peterson, J.Lexford, S.Shenker, and J.Turner. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review, Vol.38, No.2, pp.69-74, 2008.

[4] Barefoot Networks, http://www.barefootnetworks.com

[5] Twitter, http://twitter.com/

[6] Chihiro Maru et. al., "Network Failure Detection System for Traffic Control using Social Information in Large-Scale Disasters", ITU Kaleidoscope Conference 2015: Trust in the Information Society, S5.3, Dec. 2015.

[7] Akihiro Nakao, "FLARE: Open Deeply Programmable Network Node Architecture," Stanford Univ. Networki ng Seminar, October 2012. http://netseminar.stanford.edu/10_18_12.html

[8] Akihiro Nakao, "Software-Defined Data Plane Enhancing SDN and NFV", Special Section on Quality of Diversifying Communication Networks and Services, IEICE Transactions on Communications, vol.E98-B, No.1, pp.12-19, Jan. 2015.

[9] NTT DOCOMO, Tohoku University, NEC, Fujitsu, Hitachi Solutions East Japan. "Experimental challenges for dynamic virtualized networking resource control over an evolved mobile core network - a new approach to reduce massive traffic congestion after a devastating disaster", R & D from 2011 to 2012 of MIC.

[10] Koichi Ogawa, Noriaki Yoshiura. "Network Priority Control during Disasters and effectiveness Evaluation", Technical Report of the Proceedings of the Institute of Electronics, Information and Communication Engineers, vol.2014-IOT-24, no.23, pp.1-6, Feb. 2014.

[11] Koichi Ogawa, Noriaki Yoshiura. "Network operational Methods based on Geographical Infomation with Consideration of the Reliability of Communication",DICOMO 2015, 8B-2, pp.1646-1652, 2015.

[12] Yuki Kumagai, Yuto Sekino, Noriki Uchida, Yoshitaka Shibata. "Realization of The End-to-End Route Selection Method in The Disaster based on OpenFlow", The 75th National Convention of IPSJ, pp.3-337-338, Mar. 2013.

[13] Yuto Sekino, Yoshitaka Shibata, Noriki Uchida, Norio Shiratori. "Research on the Implementation of Link Switching Method in Disaster Information Network Based on OpenFlow Framework", Information Processing Society of Japan(IPSJ) SIG Technical Report, Vol.2013-DPS-154 No.49, Mar. 2013.

[14] Goshi Sato, Yoshitaka Shibata, Noriki Uchida. "Research on SDN technology based Disaster Resilient Network", The 77th National Convention of IPSJ, pp.3-187-188, Mar. 2015.

[15] Asato Edo, Satoru Izumi, Toru Abe, Takuo Suganuma. "Design and Implementation of Disaster-Risk-aware Smart Routing", DICOMO 2015, 7E-3, pp.1520-1524, July 2015.

[16] The Click Modular Router Project, http://www.read.cs.ucla.edu/click/

[17] Akihiro Nakao, Ping Du. "Application and Device Specific Slicing for MVNO", 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), Oct. 2014.

[18] Akihiro Nakao, Ping Du, Takamitsu Iwai. "Application Specific Slicing for MVNO through Software-Defined Data Plane Enhancing SDN", IEICE TRANSACTIONS on Communications Vol.E98-B, No.11 pp.2111-2120, 2015.

[19] JGN-X, http://www.jgn.nict.go.jp