Congestion Prevention Mechanism Based on Q-learning for Efficient Routing in SDN

Seonhyeok Kim, Jaehyeok Son, Ashis Talukder, and Choong Seon Hong
Department of Computer Science and Engineering
Kyung Hee University
446-701, Republic of Korea
E-mail : {kshyuk0605, sonjaehyeok, ashis, cshong}@khu.ac.kr

***Abstract*—SDN (Software-Defined Networking) has been proposed to solve problems caused by difficulties in central management, vendor dependency, and increase in complexity of network due to individual process. In current SDN, however, routing algorithm is mainly based on Dijkstra's Algorithm and the shortest Flow Path is selected to deliver packets. This may result in network congestion since bandwidth overhead is not considered when a lot of traffic enters in the network. Therefore, we propose a mechanism to prevent network congestion based on Q-learning for efficient routing in SDN. In this paper, we show the network congestion can be improved by reselecting the path and changing Flow Table using predefined threshold and Q-learning routing algorithm**

Keywords—Software Defined Networking, Q-learning, Controller

I. Introduction

Cloud, Mobile, IoT and other IT techniques has been developed and the usage of Internet has increased dramatically. As a result, many structural problems such as difficulties in central management and automation of network, vendor dependency and increase in the complexity of network due to individual process are emerged. In order to solve such problems, many techniques in terms of future Internet are being studied and developed. Especially, SDN (Software-Defined Networking) draws a great attention since it enables programmable network, central control and flexible management [1] [2]. Basically, control plane and data plane are separated in SDN to provide the ability to control and manage the network by software programming. Fig. 1 represents the reference model of SDN which is composed of three layers. Infrastructure layer takes the role of collecting network status information and sending packets. Control layer controls the devices in infrastructure layer and provides various service AP (Access Point). In addition, application layer provides application program to satisfy the demand of users [3].

In this paper, we point out the problems that Dijkstra's Algorithm used in the existing SDN face and propose the solution to solve the problems with minimization of network congestion using threshold value and Q-learning which is a kind of reinforcement learning. In this paper, we construct virtual network environment to validate the performance of the proposed algorithm compared to those used in current routing protocols.

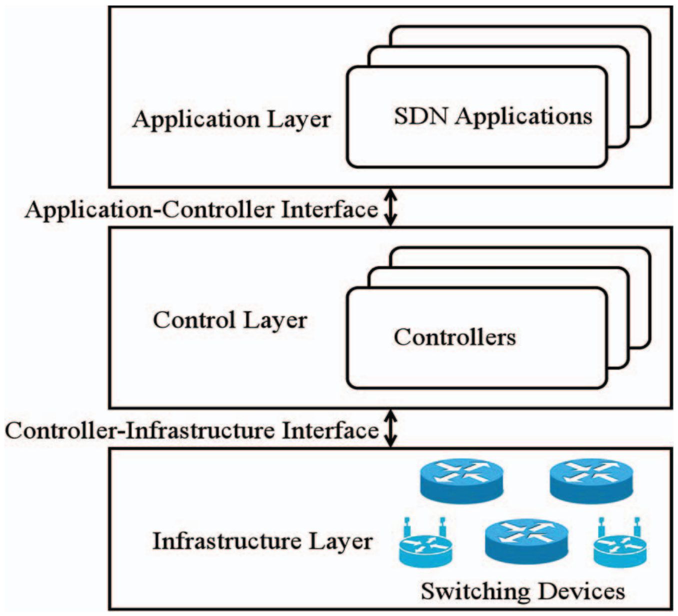


Fig. 1. SDN refernece model

The rest of the paper is constructed as follows. In section 2, we review related works. In section 3, we propose the way to minimize network congestion. Afterwards, we demonstrate the evaluation and simulation in section 4. We finally conclude this paper and explain about future works in section 5

II. RELATED WORK

A. *OpenDaylight*

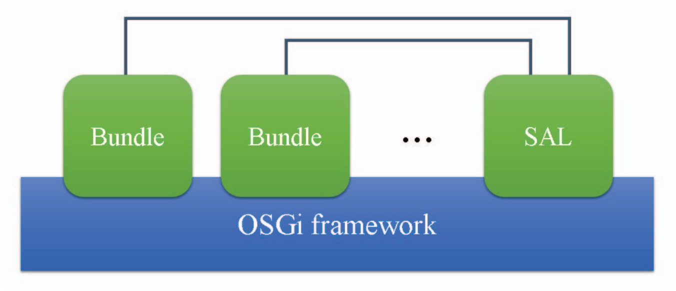


Fig. 2. OSGi framework

ODL (OpenDayLight) of Linux Foundation is open source software platform for SDN and NFV (Network Function Virtualization). Many IT companies such as Cisco, IBM and Dell are participated in ODL project to improve the quality of SDN and NFV. ODL supports OpenFlow, OpFlex and many different network device control protocol and it also has the advantage with respect to portability since it uses standardized model [4].

ODL architecture can be divided into four categories. Firstly, Network Applications Orchestrations&Services is for orchestration and management of network. Also, it provides interface between controller and application using REST(Representational Safe Transfer) API. Secondly, Controller Platform supports JAVA language-based OSGi framework. Besides, it controls Data Plane and manages protocol plugin. Thirdly, Southbound Interfaces & Protocol Plugins provides protocols like OpenFlow and NETCONF to control Data Plane through Southbound Interface. Lastly, Data Plane Elements consist of either physical or virtual network devices [4].

Among these, ODL Controller Platform is the core part in ODL. ODL Controller Platform is SDN virtualization layer providing many different protocols to control data plane as well as REST API for applications. Unlike other controllers, various protocol support is available since SAL (Service Abstraction Layer) makes decision about protocols between controllers and network devices. Moreover, it functions based on JAVA so it can be used in operating systems and devices that supports JAVA. In addition, ODL uses OSGi framework for modularization and scalability. OSGi framework is abbreviation of Open Services Gateway Initiative framework. It is JAVA-based framework that can either add or delete bundle (Application or Component) dynamically without rebooting process [5]. Fig. 2 represents OSGi framework structure.

B. *Q-learning*

Reinforcement learning is the process to find out the policy that maximizes cumulative rewards of agent. The agent acts according to current state while detecting environment and it receives reward from the environment. Fig. 3 represents the process of Reinforcement learning.

Q-learning is one of the reinforcement learning technique used to find out the optimal action-selection policy based upon Markov decision process [6] [7]. The feature of Q-learning is in its capacity to choose between immediate rewards and delayed rewards. The agent, at initial stage, selects x_t while observing one of the state x_t and its vector. One of the action u_t is then applied to x_t. As the process moves from x_t state to x_{t+1}, the agent receives reinforcement r(x_t, u_t). The main purpose of Q-learning is to find out the sequence of actions that maximizes the sum of future reinforcements in order to know the shortest path from a source to a destination.

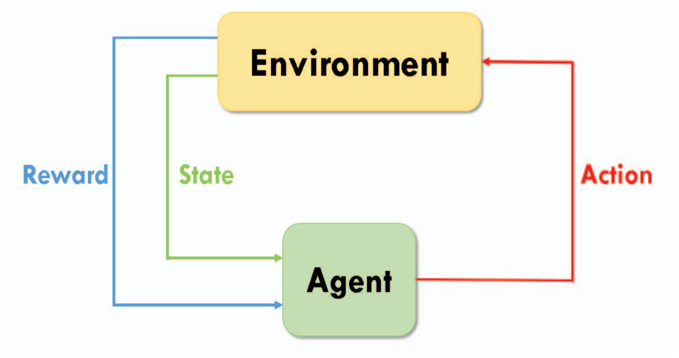


Fig. 3. Reinforcement learning

C. *mininet*

Mininet, distributed from Stanford university, is OpenSource project which makes is possible to construct virtual network on personal computers or laptops for test purpose. In other words, it is a network emulator that can create network using virtual hosts, switches, controllers, and links [8]. Also, mininet supports linux-based host and OpenFlow to switches for SDN. It is very useful tool for initial developmnt debug and evaluate network environment since we can easily modify information such hosts, switches, links and so on.

III. Proposal

A. *Problems of routing using Dijkstra's algorithm in ODL.*

If we consider the one of SDN controller, OpenDayLight, it uses routing module based on Dijkstra's algorithm [9]. Dijkstra's algorithm is an algorithm to calculate the shortest path between source and destination in a given graph. In ODL, the shortest path between source and destination is calculated first and Flow Table in OVS (OpenVSwitch) is modified based on the path information using Southbound API which is OpenFlow protocol. When traffic needs to be delivered it follows the predefined path in the Flow Table [10].

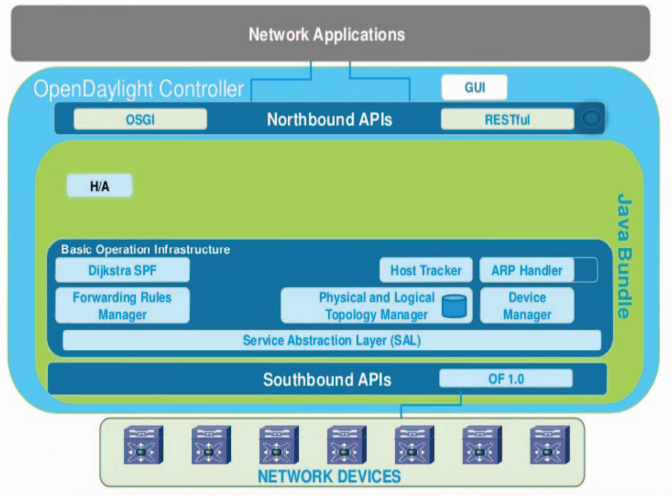


Fig. 4. Dijkstra SPF module in OpenDaylight

Fig. 4 shows ODL network architecture and Dijkstra Shortest Path First (SPF) module. It also shows that OVS can be controlled by OpenFlow protocol. However the existing ODL which uses Dijkstra's algorithm does not consider the changes in bandwidth of each link when selecting the routing path. So, if there is sudden increase in traffic, network congestion may occur due to the inconsideration of bandwidth overhead, which leads to decrease in packet transmission rate. In other words, packet is only forwarded through the path that was defined at the initial stage. This causes network performance degradation.

In this paper, we solve the problems mentioned in the previous clause using threshold value defined in controllers and bandwidth detection. If the exceeded amount of traffic is in the network while detecting bandwidth, the controller searches for optimal routing path to prevent the network congestion by traffic prediction using Q-learning routing algorithm. The updated optimal path is then delivered to OVS through OpenFlow and traffic flows are scattered as Flow Table is modified in a way to prevent the network congestion.

B. Congestion prevention scenario

Fig. 5 represents the network topology to test the proposed Q-learning based network congestion prevention algorithm. In this section, we describe the scenario of the proposed mechanism to prevent network congestion using the network topology shown in Fig. 5.

When a large amount of traffic needs to be delivered from Host1 to Host6, it follows OVS1 to OVS3 to OVS6 path based on Dijkstra's algorithm. If there is another trial to transmit data from Host3 to Host7, a serious problem that data flows are over-crowded at the path between OVS3 and OVS6. This means that traffic is increased in the path between OVS3 and OVS6, which leads to bandwidth overhead so network congestion occurs.

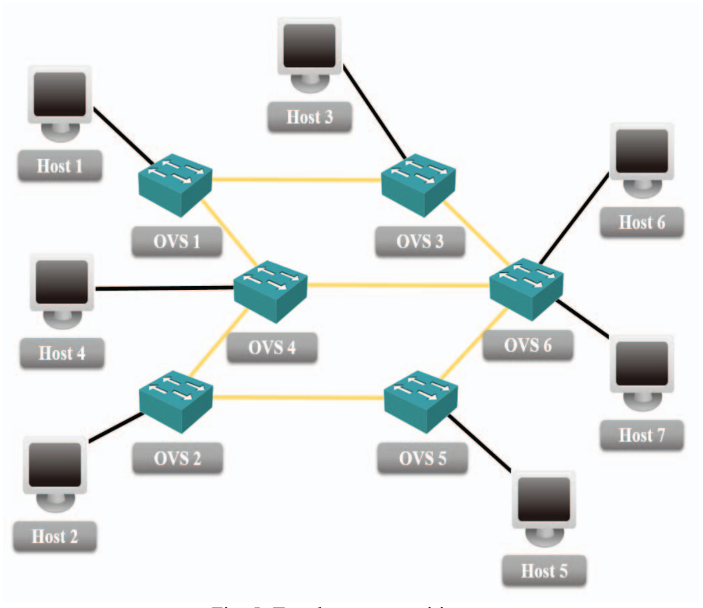


Fig. 5. Topology composition

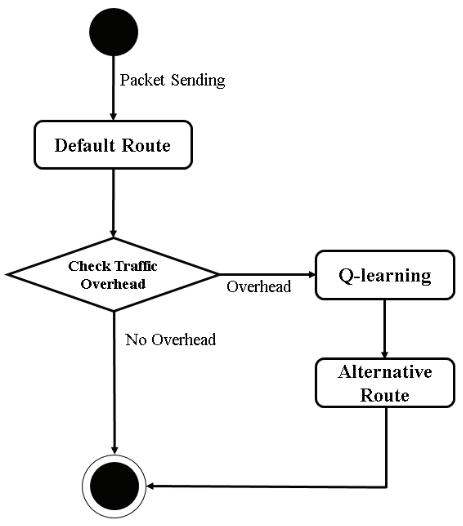


Fig. 6. State diagram of packet transmission

In this paper, we set up threshold value for every link to prevent the network congestion. When traffic exceeds the pre-defined threshold value, controller searches for a new routing path based on Q-learning algorithm. Furthermore, controller modifies Flow Table of each OVS using OpenFlow. Routing information in the topology becomes OVS1 to OVS4 to OVS6 and data are transmitted through the path. The proposed mechanism makes it reliable to maintain efficient network status.

Fig. 6 is a diagram that represents the proposed mechanism. At first, packets are transmitted through the route based on Dijkstra's algorithm. While packets are being delivered, the controller periodically checks if there is network overhead. If so, an alternative route is created for an efficient packet delivery using Q-learning algorithm.

C. Q-learning routing algorithm

In this clause, we deal with Q-learning routing algorithm to cope with network congestion considering network overhead in case where a large amount of traffic suddenly occupies the network. Algorithm 1 represents Q-learning routing algorithm as pseudo code.

First of all, CB (CurrentBandwidth), gamma and Iteration are set to be 800Mbit, 0.7 and 10 respectively. Besides, Q-Matrix sets to be 0 while R-Matrix sets to be reward value according to network composition [6].

In this algorithm, controller measures the bandwidth of network edge. In case where PB(PredictedBandwidth) becomes more than 80% of CB it is probable that overhead can occur at the edge so R-Matrix value is changed. R-Matrix value obtains weight of 30 if the next node is the destination. Otherwise it obtains weight of 20. Reward value is modified by the updated R-Matrix value. A new optimal path can now be calculated by applying the modified value to Q-learning algorithm to prevent network congestion.

Algorithm 1. Pseudo code for Q-learning routing algorithm

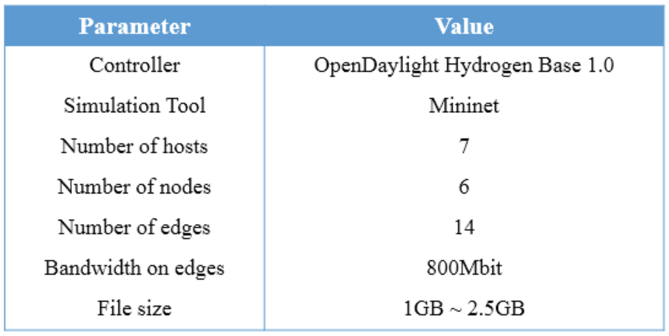
```
 Input:  Edge, Vertex, G=(V, E), PB(PredictedBandwidth),
         CB(CurrentBandwidth), Reward, gamma, Iteration,
         Q-Matrix, R-Matrix, TI(TopologyInformation)
Output:  Q-Matrix, route

 1: Initialization
 2:   CB(CurrentBandwidth) ← 800Mbit,
 3:   gamma ← 0.7, Iteration ← 10, Q-Matrix ← 0
 4:   Update the Reward using TI(TopologyInformation)
 5:
 6: while(Traffic Flow)
 7:     R-Matrix ← Reward
 8:     for all e ∈ G.edges()
 9:         if (PB > CB * 0.8) then
10:             e ⇒ V_src , V_dst
11:             if ( V_dst == Goal state ) then
12:                 R[V_src ][ V_dst ] value is 30
13:             else
14:                 Find V adjacent V_src
15:                 R[V_src ][V] value is 20
16:             endif
17:         endif
18:     endfor
19:
20:     while(Iteration)
21:         Q(state, action) = R(state, action) + gamma
22:                            * Max[Q(next state, all actions)]
23:     endwhile
24:
25:     Update Q-Matrix
26:     Update route
27: endwhile
```

IV. Evaluation

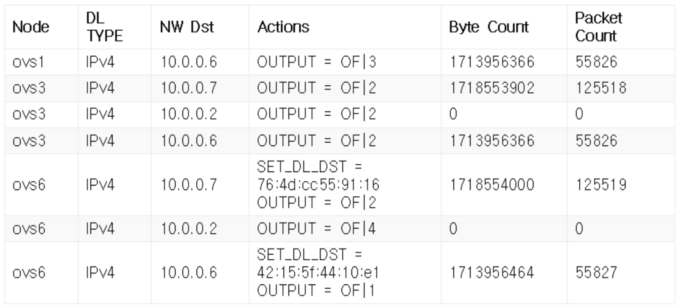
A. Simulation environment

In this clause, we define parameters and environments for simulation to evaluate the validation of proposed algorithm Fig. 7 shows the values of parameters used in the simulation and virtual network system environment that uses mininet. The virtual network environment consists of one SDN controller, 7 hosts, 6 OVS and 14 edges that connects the hosts and OVSs.



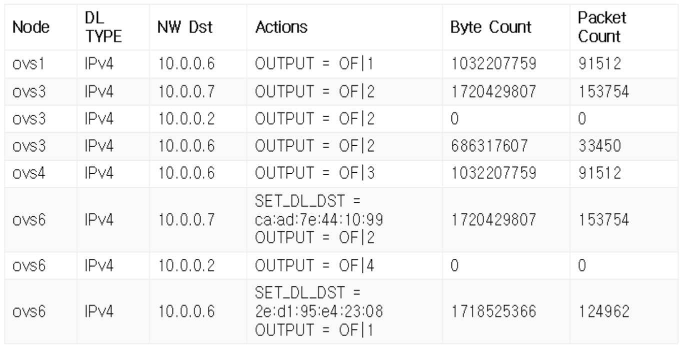
Parameter	Value
Controller	OpenDaylight Hydrogen Base 1.0
Simulation Tool	Mininet
Number of hosts	7
Number of nodes	6
Number of edges	14
Bandwidth on edges	800Mbit
File size	1GB ~ 2.5GB

Fig. 7. Simulation environment



Node	DL TYPE	NW Dst	Actions	Byte Count	Packet Count
ovs1	IPv4	10.0.0.6	OUTPUT = OF|3	1713956366	55826
ovs3	IPv4	10.0.0.7	OUTPUT = OF|2	1718553902	125518
ovs3	IPv4	10.0.0.2	OUTPUT = OF|2	0	0
ovs3	IPv4	10.0.0.6	OUTPUT = OF|2	1713956366	55826
ovs6	IPv4	10.0.0.7	SET_DL_DST = 76:4d:cc55:91:16 OUTPUT = OF|2	1718554000	125519
ovs6	IPv4	10.0.0.2	OUTPUT = OF|4	0	0
ovs6	IPv4	10.0.0.6	SET_DL_DST = 42:15:5f:44:10:e1 OUTPUT = OF|1	1713956464	55827

Fig. 8. Result of data transmission in existing ODL



Node	DL TYPE	NW Dst	Actions	Byte Count	Packet Count
ovs1	IPv4	10.0.0.6	OUTPUT = OF|1	1032207759	91512
ovs3	IPv4	10.0.0.7	OUTPUT = OF|2	1720429807	153754
ovs3	IPv4	10.0.0.2	OUTPUT = OF|2	0	0
ovs3	IPv4	10.0.0.6	OUTPUT = OF|2	686317607	33450
ovs4	IPv4	10.0.0.6	OUTPUT = OF|3	1032207759	91512
ovs6	IPv4	10.0.0.7	SET_DL_DST = ca:ad:7e:44:10:99 OUTPUT = OF|2	1720429807	153754
ovs6	IPv4	10.0.0.2	OUTPUT = OF|4	0	0
ovs6	IPv4	10.0.0.6	SET_DL_DST = 2e:d1:95:e4:23:08 OUTPUT = OF|1	1718525366	124962

Fig. 9. Result of data transmission applying Q-learning algorithm

In the simulation, we use the network topology shown in Fig. 5 The bandwidth of each edge is fixed with 800Mbit so that data can be transmitted under the same circumstance. We obtain the simulation result according to the size of data, 1GB, 1.7GB and 2.5GB

B. Data Transmission Result in ODL

Fig. 8 and Fig. 9 represents the screen of controller that shows the difference between ODL and the proposed algorithm when 1.7GB of Data are sent from Host 3 to Host 7 while 1.7GB of data are transmitted from Host 1 to Host 6 [11].

In Fig. 8, it is shown that although there occurs network overhead data are sent through OVS1 to OVS3 to OVS6 at first stage.

Fig. 9 shows the captured screen of controller which applies the proposed algorithm that recognizes changes in bandwidth. At first, data transmission follows the same order of path which is from OVS1 to OVS6 via OVS3. In case of overhead between Host 3 and Host 6, controller recognizes it and the path is reconfigured as optimal so data are sent from OVS1 to OVS6 via OVS4.

C. Simulation result

In this clause, we mainly deal with comparison results to prove the efficiency of the proposed algorithm under the virtual network environment.

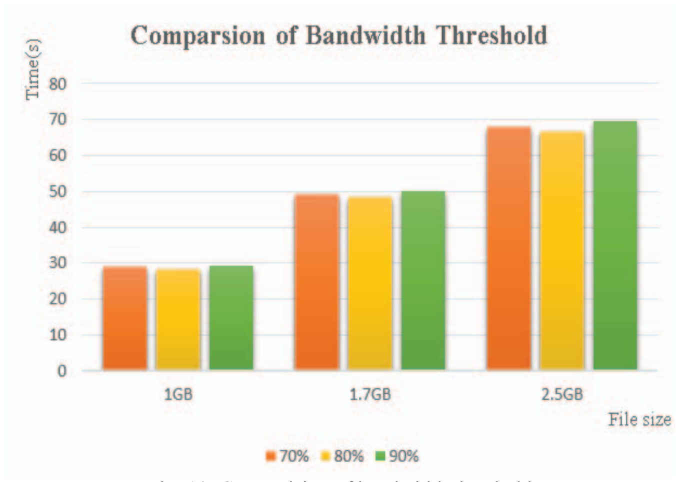


Fig. 10. Comparision of bandwidth threshold

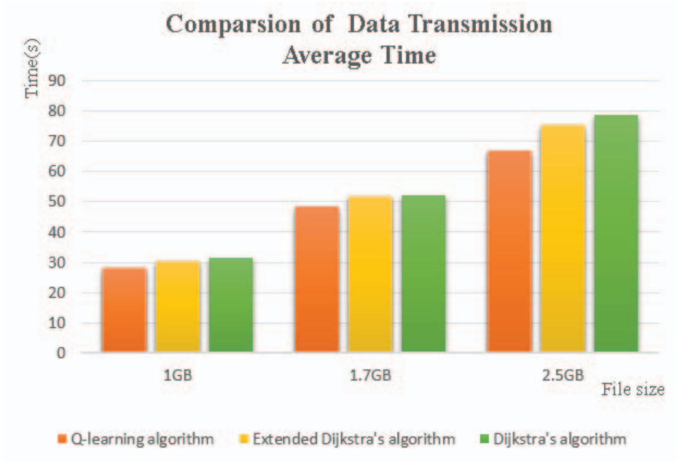


Fig. 11. Comparison of data transmission average time

Fig. 10 shows the result of comparison of bandwidth threshold. The result implies that the bandwidth threshold should be set to 80% to cope with network congestion. In this simulation, we vary the size of the file to get more validate results.

In addition to the result in relation to the bandwidth threshold, we compare the data transmission time according to three different types of algorithms [10]. As it is mentioned, Fig. 11 shows the data transmission comparison results among Dijkstra's algorithm, Extended Dijkstra's algorithm and the proposed Q-learning routing algorithm. The result demonstrates that there is no big difference in case of 1GB but the proposed algorithm shows better performance result as the size of the file increases compared to other algorithms.

V. Conclusion

In this paper, we defined the problem of existing ODL, which does not consider changes in bandwidth. As to address the problem, we proposed network congestion prevention mechanism based on Q-learning algorithm. In order to prove the efficiency of the proposed algorithm, we performed simulations and the results shows that the proposed algorithm can solve network congestion problem.

However, it is only limited to a certain environment that has fixed traffic generation rate and the size of bandwidth. Therefore, more flexible and scalable researches to deal with such problems are required to cope with the breakthrough of IT industry. Furthermore, the proposed algorithm is required to be improved to satisfy the demand of users.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (B0190-15-2013, Development of Access Technology Agnostic Next-Generation Networking Technology for Wired-Wireless Converged Networks). *Dr. CS Hong is the corresponding author.

References

[1] D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. uhling, "Software-Defined Networking: A Comprehensive Survey," proceecding of the IEEE, Vol. 103, pp. 14-76, January 2015.

[2] B.A.A. Nunes, M. Mendonca, N. Xuan-Nam, K. Obraczka, and T. Turletti, "A survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," Communications Surveys & Tutorials, IEEE, Vol. 16, pp. 1617-1634, February 2014.

[3] X. Wenfeng, W. Younggang, F.H. Chuan, D. Niyato, and X. Haiyong, "A Survey on Software-Defined Networking," Communications Surveys & Tutorials, IEEE, Vol. 17, pp. 27-51, June 2014.

[4] Z.K. Khattak, M. Awais, and A. Iqbal, "Performance evaluation of OpenDaylight SDN controller," Parallel and Dstributed Systems (ICPADS), 2014 20th IEEE Conference on, pp. 671-676, December 2014.

[5] C. Junqing, H.Linpeng, D. Siqi, and Z. Wenjia, "A Formal Model for Supporting Frameworks of Dynamic Service Update Based on OSGi," Software Engineering Conference (APSEC), 2010 17th Asia Pacific, pp. 234-241, November 2010.

[6] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Q-learning based Congestion-aware Routing Algorithm for On-Chip Network," Networked Embedded Systems for Enterprise Applications (NESEA), pp. 1-7, December 2011.

[7] S. Petrangeli, M. Claeys, S. Latre, J. Famaey, and F. De Turck, "A Multi-Agent Q-Learning-based Framework for Achieving Fairness in HTTP Adaptive Streaming," Network Operatoins and Management Symposium (NOMS), pp. 1-9, May 2014.

[8] B. Lantz, B. Heller, and N. Mckeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," ACM SIGCOMM Workshop on Hot Topics in Networks, pp. 1-6, October 2010.

[9] J.L. Izquierdo-Zaragoza, A. Fernandez-Gambin, J.J. Pedreno-Manresa, and P. Pavon-Marino, "Leveraging Net2Plan planning tool for network orchestration in OpenDaylight," Smart Communications in Network Technologies (SaCoNet), pp. 1-6, June 2014.

[10] J. Jehn-Ruey, H. Hsin-Wen, L. ji-Hau, and C. Szu-Yuan, "Extending Dijkstra's shortest path algorithm for software defined networking," Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific, pp. 17-19, September 2014.

[11] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, Vol. 38, pp. 69-74, April 2008.