Journal of Theoretical and Applied Information Technology
15th February 2020. Vol.98. No 03

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN EFFICIENT ROUTING METHOD IN SDN FOR SMART INTELLIGENT SYSTEMS

[1]HO-JIN HEO, [1]NAMGI KIM

[1]Department of Computer Science and Engineering, Kyonggi University, Suwon 16277

E-mail: {h_hojin, ngkim}@kgu.ac.kr

*Corresponding Author: Namgi Kim (ngkim@kgu.ac.kr)

ABSTRACT

There are lots of multimedia services such as YouTube and Netflix in the Internet. The multimedia services through the Internet will continue to grow. Most of these multimedia services generally stream a content to many users. In these type services, the multicast transmission mode can efficiently deliver the content to multiple subscribers. Especially in SDN (Software-Defined Network), multicast mode can be easily adopted because the centralized controller sets up all routes using multicast tree with global network information. However, the construction of multicast tree is an NP (Non-deterministic Polynomial-time) problem and is hard to make the optimal multicast tree in the real world. Therefore, in this paper, we propose a heuristic way to generate a multicast tree using DQN (Deep-Q-Network) which is a type of reinforcement learning in machine learning field. Through the experiment, we show that the performance ratio of the proposed algorithm is 1.21 with the topology of 10 nodes and it generates the multicast tree better than the previous heuristic algorithms such as TM (Takahashi and Matsuyama) algorithm.

Keywords: *DQN, Reinforcement Learning, Multicast, SDN*

1. INTRODUCTION

Currently, various multimedia services are provided on the Internet based on videos. Such Internet video services include short-term services like YouTube, long-term like Hulu, and Internet video to TV services like Netflix. In addition, various multimedia services such as live Internet video services, online video purchase and rental services, webcam viewing services, and web-based video surveillance services are provided through the Internet. According to the white paper of Internet traffic forecast and trends of Cisco [1], Internet video traffic will account for 82% of all Internet traffic by 2022. Particularly, if the 4K UHD and 8K UHD video markets flourish and surpass HD, one-source-multi-use services, which transmit one content to many users through the Internet, will also increase exponentially [1].

The transmission mode that can most widely be used to deliver one content to many users is multicast. In the multicast transmission mode, network resources can be utilized much more efficiently, compared to the method of delivering contents to multiple users on the application layer by using the unicast transmission mode, because the data that started from a single node is copied onto a network layer and delivered to many users. However, the conventional multicast mode presents the problem that a global-optimal multicast tree cannot be constructed, and local-optimal multicast trees are constructed instead because routers have distributed topology information [2–6]. To resolve this problem, studies on multicast in a Software-Defined Network (SDN) have emerged.

In the SDN, a multicast tree can be constructed with global topology information through an SDN controller. However, the problem of constructing a minimum cost multicast tree is like the problem of constructing a Steiner tree, and this is NP-hard. To resolve this, various approximation techniques have been researched, but no study has yet adopted a Deep Q-Network (DQN).

Therefore, in this paper, we investigates a method of generating a multicast tree efficiently by using a DQN. The structure of this paper is organized as follows. Section 2 describes the background and related works that were basically required to perform this study. Section 3 introduces a method that can construct a multicast tree effectively using a DQN. Section 4 introduces the experimental environment and analyzes the results of various experiments performed in that environment. Finally, Section 5 provides a conclusion based on these results and describes a future project.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. RELATED AND PRIEVIOUS WORKS

In the unicast mode, a packet is transmitted from a source node to a destination node in a one-to-one fashion. Therefore, when one source node and multiple destination nodes exist, the packets are transmitted as many as times as the number of destination nodes. By contrast, in the multicast mode, only one packet is forwarded from the source node, and in the process of delivering the packet, the router that has to forward the packet to two or more links copies and forwards the packet. Consequently, it prevents the same packet from being transmitted multiple times through the same link. Therefore, in a multimedia service that characteristically sends packets from one node to multiple nodes, the multicast transmission mode offers the advantage of using the network resources more efficiently.

The method of implementing the multicast mode in a distribution environment has been proposed for a very long time. The multicast routing protocols used in the existing Internet environment include PIM (Protocol-Independent Multicast) [7], IGMP [8], DVMRP [9], and MBGP [10]; among these, PIM protocol is the most widely used. In addition, there are variants of PIM such as PIM-SM (PIM Sparse Mode) [11], PIM-DM (PIM Dense Mode) [12], Bidir-PIM [13], and PIM-SSM (PIM-Source multicast) [14]. However, these routing protocols are based on the distributed routing algorithm, in which each router gathers routing information independently and calculates a final route. However, in the existing Internet environment, which uses distribution environments, because respective routers can hardly gather all global routing information, a constructed multicast tree becomes a local-optimal multicast tree, which is actually far from being optimal. Furthermore, because a variety of heterogeneous routers exist independently, routers that do not support the multicast mode also exist, thereby reducing the effect of the multicast transmission mode.

With the SDN being proposed in recent years, respective routers do not calculate the routes independently. Instead, in every autonomous system, a centralized controller gathers all routing information, constructs a routing tree, and sends the result to a switch. An SDN is divided into the control plane and the data plane. A route calculation occurring on the control plane is performed using the centralized method at the SDN controller, and the result is later forwarded to a sub-level switch by using a protocol such as OpenFlow [15]. Therefore, the controller can perform centralized network controls such as forwarding control, topology and resource state management, and routing control for packets based on the global view of a network state. Furthermore, according to upper-level application service or policy requirements, differentiated forwarding and packet processing rules are determined and sent down to a sub-level SDN switch. Hence, the network operates in a flexible manner with the aid of software [16][17]. Therefore, a multicast tree can be calculated more effectively in such an SDN environment. Furthermore, because the SDN switch basically supports the packet copy functionality, the multicast transmission mode can be used more actively.

The algorithms that have implemented the multicast mode can be largely classified into source-based tree algorithms and group-shared tree algorithms: the former transmits a packet by constructing a Shortest Path Tree (SPT) from a source node to a destination node; and the latter constructs a sharing point called a Rendezvous Point and transmits a packet to a destination node via the Rendezvous Point. The source-based tree algorithm can create an optimal path between a source node and a destination node, and minimize the network waiting time. However, the source-based tree algorithm has to maintain the path information for each source node at the router, and a resource problem will inevitably occur in a network that has numerous source nodes and numerous groups. Typical source-based tree multicast protocols include MOSPF [18], DVMRP [9], and PIM-DM [12]. By contrast, although the group-shared tree algorithm does not form an optimal path, it has an advantage that not so many resources are needed. The group-shared tree algorithm needs a multicast dedicated router that handles the Rendezvous Point, and the corresponding router's location has a large impact on the efficiency of a generated multicast tree. Typical group-shared tree multicast protocols include PIM-SM [11] and CBT [19]. However, the source-based tree algorithm and the group-shared tree algorithm all still have the same problem that a globally optimal multicast tree cannot be generated because the routers possess local information only.

A multicast tree can be classified as either a single multicast tree that has a single source node, or a multiple multicast tree that has multiple source nodes. This study considers the generation of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

single multicast tree only to firstly reduce the complexity of problem.

The multicast tree generation problem in theory eventually becomes the Steiner tree generation problem [20]. Because the Steiner tree generation problem is NP-hard and very difficult to solve, many approximation algorithms exist for its solution. The most typical single multicast algorithm is the TM algorithm [21]. In the TM algorithm, a graph $G = (V, E)$, a source node src_i, and a destination node set D_i are given as inputs, and multicast tree $T_i = (V_i, E_i)$ is the output. A multicast tree T_i consists of a set of multicast nodes V_i and a set of multicast links E_i. The TM algorithm is the mostly widely used among the approximation algorithms for multicast tree construction, and the performance ratio value is known to be 2. The performance ratio is a measure of how close the constructed multicast tree and the optimal multicast tree are. The performance ratio is the total weight of constructed multicast tree over the total weight of the optimal multicast tree. This study uses a machine learning method called reinforcement learning to generate a multicast tree. As shown in Fig. 1, reinforcement learning is a method in which an agent takes an action from among the selectable actions in the current state of the environment observed, that maximizes the rewards, and then repeats the action until the termination state is reached. The ultimate goal of a reinforcement learning algorithm is to let an agent decide actions in a way that maximizes the cumulative rewards until the termination condition is reached. The environment handled in the reinforcement learning process is given as a Markov Decision Process (MDP) [22]. In reinforcement learning, correct input and output pairs are not given, unlike supervised learning. Furthermore, one of the important factors in reinforcement learning is selecting between exploration and exploitation. An agent chooses whether to take actions based on learned knowledge, or to take actions by using knowledge other than that already learned.

This study uses a DQN among the reinforced learning methods. A DQN is a variant of Q-learning [23], and a model-free reinforcement learning technique. In Q-learning, the value function is expressed by $Q(s, a)$, and in given state S, it is a learned policy to select action a. The principle of a DQN is that because there is a Q-table that has a result value for each Q function, the optimal rewards are known for all state and action pairs. If the amount of data increases in a Q-learning system that uses a Q-table, the size of the Q-table increases and consequently, learning is difficult when there are too many states and actions. A DQN is a variant technique that has solved this problem. The DQN solved the problem of Q-learning being unable to process large data by applying a neural network instead of a Q-table. When a nonlinear function approximator such as a neural network is used in reinforcement learning, unstable or divergent problems arise, and the DQN solves the non-stationary problem by using a method of updating a target network periodically. Furthermore, a data correlation problem generally exists in reinforcement learning, but the DQN solves the correlation problem between data by training the samples in a batch method by using relay memory [24].

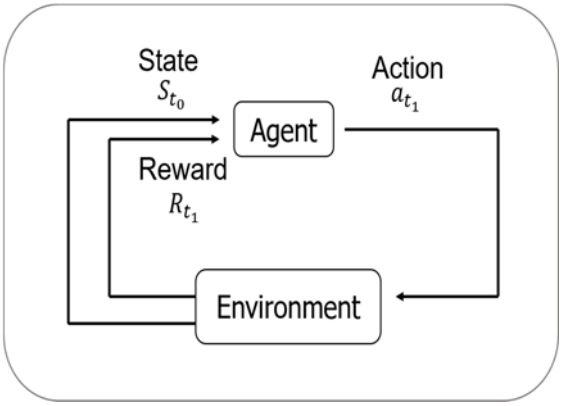


Figure 1: Reinforcement learning

DQN is a variant of Q-learning, and this technique ensures a number of things: Firstly, that the optimal policy can be learned in a state S; secondly, that large learning data can be handled easily; and thirdly, that the data correlation problem is solved. In the case of this particular study, an enormous amount of learning data exists. The number of cases that can be assigned pairs of one source node and multiple destination nodes is provided by Eq. (1), and in the case of 10 nodes, there are 5,110 source node and destination node pairs.

$$\mathrm{N} \times \sum_{n=1}^{N-1} C(N-1, n) \quad (1)$$

Here, the amount of data to be learned increases exponentially according to the number of links existing in the topology. Therefore, it has been ascertained that an extremely large amount of learning data exists in this study. Furthermore, it can be said that the data correlation between

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

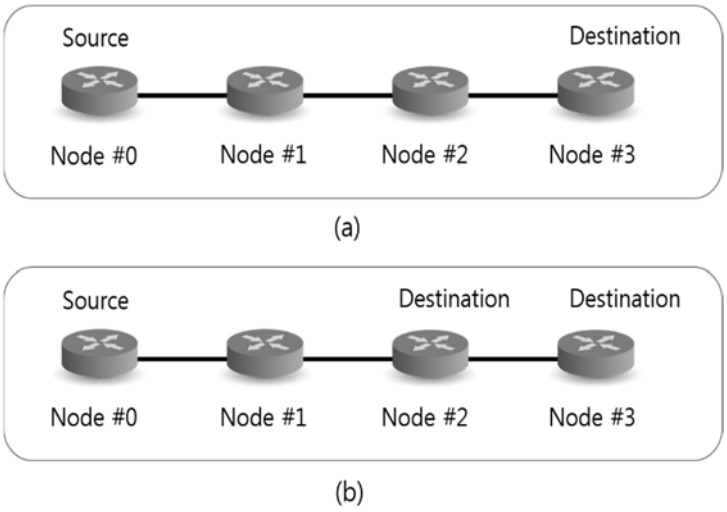


Figure 2: Example topology of creating correlated learning data

learning data is very high. As shown in Fig. 2, two types of multicast group can exist in the same topology. The multicast group consists of source node #0 and destination node #3 in Fig. 2(a) and source node #0 and destination node #2 and #4 in Fig. 2(b). In both cases, if a packet has reached the node #3, the termination state is reached by going through a same path. Therefore, it is very appropriate to use the DQN in this study.

3. SINGLE MULTICAST TREE CONSTRUCTION USING DQN

The proposed method of this study is a multicast routing tree generation method using a DQN in an SDN environment. A multicast tree should be constructed for multicast routing, and optimal multicast tree construction is an NP-hard problem. Therefore, this study proposes an algorithm that generates a multicast tree by using a DQN. The proposed method largely consists of three processes: the method of expressing the network situation as an MDP, the process of learning the situation expressed as an MDP, and routing through the value function learned.

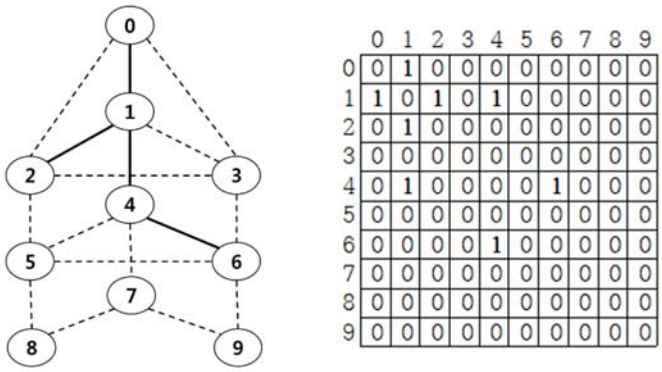
In an SDN, a controller can gather status information from multiple switches existing in a domain, and after calculating the route in a global view based on the gathered information, it can send it down to the respective switches. Since the controller can know globally the status regarding the whole network such as network topology and packet flow in the domain, it can perform the DQN learning for multicast tree construction.

The process of a multicast service can be understood as the process of ensuring a packet reaches multiple destination nodes from a single source node. In this process, the case of all packets reaching the destination nodes becomes the terminal state of the DQN. Therefore, the process of learning a multicast tree can be expressed as a finite MDP. The finite MDP is suitable for the application of Q-learning, whose goal is to find an optimal action-selection policy. However, conventional Q-learning is not suitable for learning in the real world which can have a very large number of states. Therefore, this study applies the DQN, which compensates for the disadvantages of Q-learning with the action value function. The DQN needs a state, an action, and a reward to learn a network, and the DQN agent becomes the SDN controller in the SDN environment.

In the proposed method, when n nodes exist, the state is expressed as a state matrix of $N \times N$. Once the source node and the destination nodes, which form a service group for multicast, are determined and the packets are transmitted, the current state is expressed in the $N \times N$ state matrix according to the moving path of the packet at each particular time step. Fig. 3 shows an example of expressing a matrix for the state in which a packet sent from the source node #0 has reached node #2 and node #6 in a network having 10 nodes.

The DQN agent performing as the controller selects an action that has the highest reward among many selectable actions through the network based on the current state. Here, the link that the packet passes through and the node that the packet is

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Figure 3: Example of state matrix for a topology

transmitted to are expressed in the $N \times N$ action matrix. When going through the policy, the information for all selectable actions must be expressed, and the selectable actions, i.e. selectable links, exist differently depending on the path that the packet passed through. For this, the selectable actions should be expressed variably but because the size of the output layer is fixed due to the structure of the neural network, the actions should be set according to the output layer.

Therefore, when there are N nodes, as many selectable links exists as $N(N\text{-}1)$, and this can be expressed with an $N \times N$ action matrix. The output layers of a neural network are shown by $N \times N$, and using the information about links that are impossible to select in certain network topologies, such links are thus not allowed to be selected as actions, through the $N \times N$ impossible link matrix. The SDN controller adds the non-selectable link information as a bias of the neural network through the adjacent matrix of current topology, so that the non-connectable links cannot be selected. As a result, the unconnected links are not selected as actions in the policy of the proposed method.

The proposed method grants a reward only when all packets have reached the destination node. This ensures the shortest path by discount ratio. The discount ratio is calculated as γ^t at each time step t, and the final reward is determined by multiplying the discount ratio with the reward R. The final reward is expressed as $\gamma^t \times R$, meaning that after multiplying the discount factor γ by itself t times, it is multiplied with the reward R. Because γ consists of a value between zero and one, and R is a constant larger than 0, the final value of the reward decreases as the time step t increases.

The agent, which is the SDN controller, identifies the current network state by observing

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Previous State

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Current State

Figure 4: Example of data correlation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

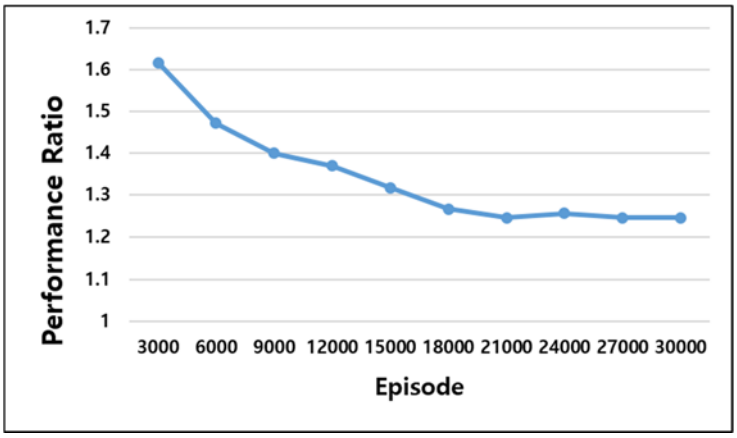


Figure 5: Performance ratio in 70% of learning cases up to 30,000 episodes

the current network environment, and selects an action through the DQN. The reward is received for the selected action, and this process is repeated until the terminal state is reached.

The correlation between previous state and current state is very high when a packet goes through a node at every time step, as shown in Fig. 4. Furthermore, when the network learning is performed, the problem of the target network becoming non-stationary is same as the problem that occurs in Q-learning. To resolve the data correlation problem, the DQN gathers the training data in the learning loop, and after saving the data in the replay buffer, it performs the training by extracting data randomly in a mini-batch format. Therefore, the non-stationary target network problem is solved by the following method: After the DQN generates a target network, it copies the training network every time a certain time step is passed.

The proposed method performs the routing through the learned model after completion of learning. The whole process of routing is as follows. First, when a single multicast service is created, and a packet is generated, the corresponding information is forwarded to the controller. The SDN controller constructs a multicast tree that the DQN learned in advance; based on this, the switches send the flow table update command. The packet is sent to the destination node via a switch that possesses the updated flow table.

The environment used in the experiments for the performance evaluation of the proposed method is shown in Table 1. In the experiments, the number of nodes was set to 10, and the number of links was randomly selected from 1 to 5 at each node. In the experiments, TensorFlow 1.2.1 GPU version was used for the DQN learning, and two layers of neural networks were used while the ReLU function was used as the activation function. The number of destination nodes was set to between 2 (minimum) and 8 (maximum) since it is from a single multicast. In the DQN, a tree without a loop was generated.

The proportion of learning data was set to 30%, 50%, and 70% of the total route cases, respectively. When the proportion of learning data is given, the history is created by selecting an action through the DQN with a source node and destination node pair for each learning case. Furthermore, when a multicast tree containing all destination nodes is constructed, one episode is completed. By performing such an episode a certain number of times, the value function is generated.

In the case of using 70% of learning cases, the experiment was performed from 3,000 episodes to 30,000 episodes in incremental steps of 3,000. The experimental results are shown in Fig. 5, and after going through 21,000 episodes, it was observed that the performance ratios were all similar.

4. EXPERIMENTS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

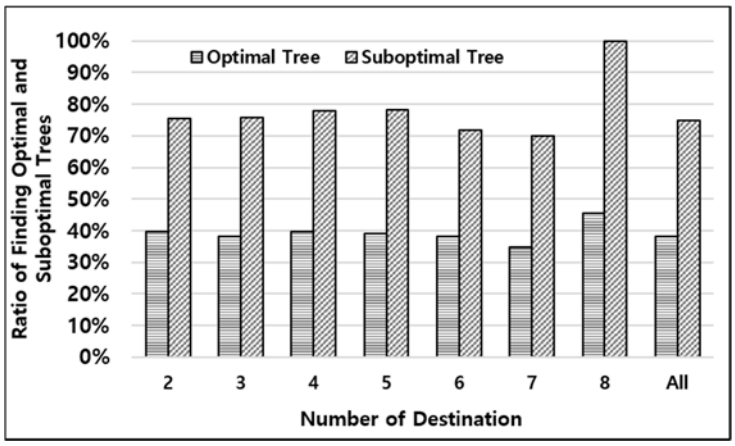


Figure 6: Construction ratio of optimal and suboptimal trees in 70% of learning cases

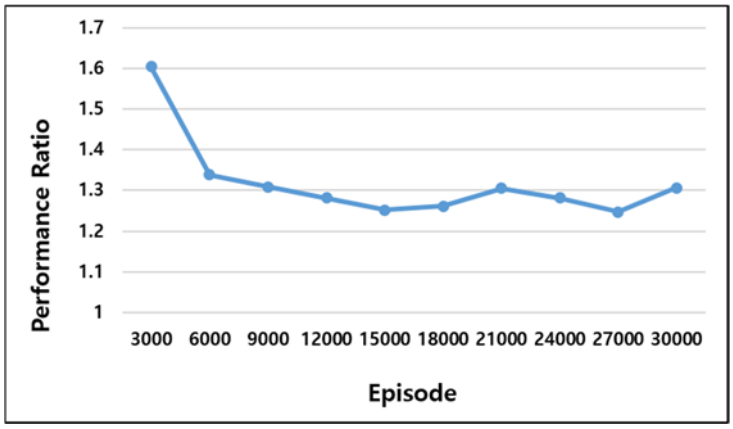


Figure 7: Performance ratio in 50% of learning cases up to 30,000 episodes

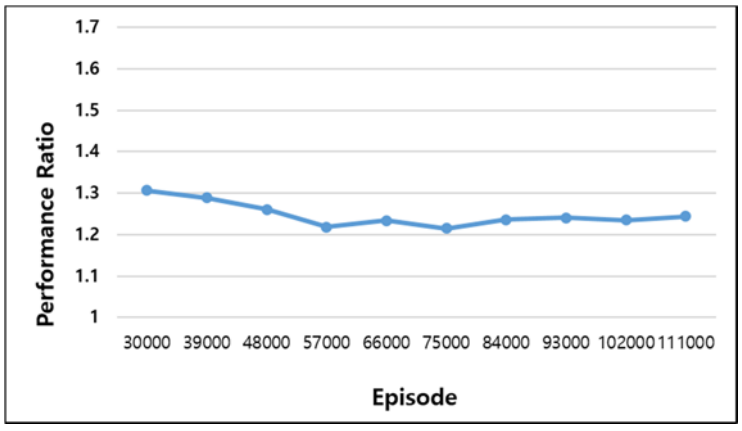


Figure 8: Performance ratio in 50% of learning cases up to 111,000 episodes

In the case of 21,000 episodes in 70% of learning cases, the average performance ratio was 1.24. Fig. 6 shows the proportion of optimal length trees found according to the number of destinations using the value function that learned the episode 21,000 times, and including the suboptimal length cases of a tree that is one more link in length than the optimal tree. On average, 38% of optimal trees

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

were found, and it was confirmed that 75% were found on average when trees were included that have the additional cost of one more link than the optimal tree.

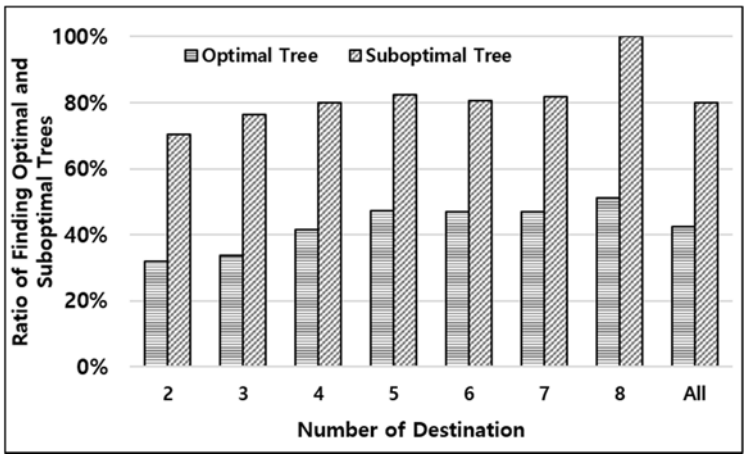


Figure 9: Construction ratio of optimal and suboptimal trees in 50% of learning cases

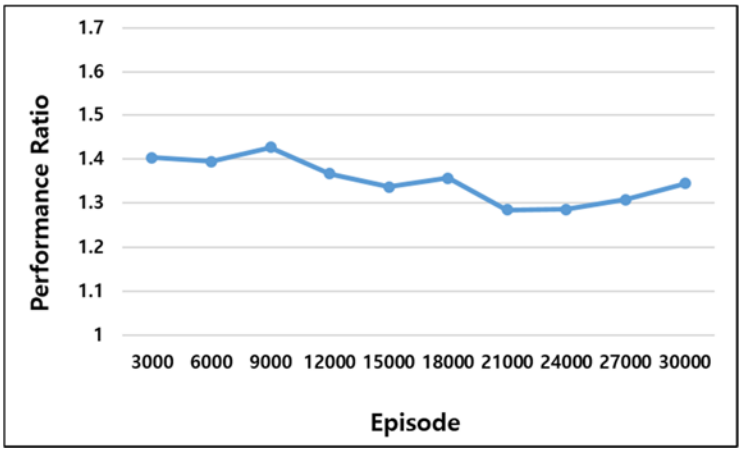


Figure 10: Performance ratio in 30% of learning cases up to 30,000 episodes

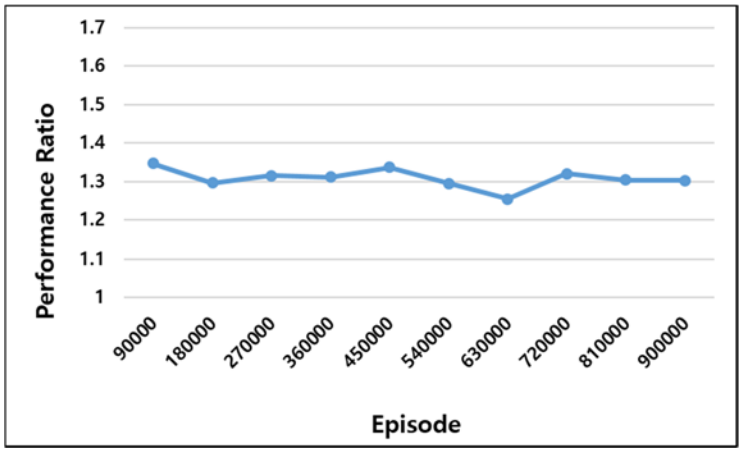


Figure 11: Performance ratio in 30% of learning cases up to 900,000 episodes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Experimental parameters

Parameters	Values
Node	10
Number of Source Node	1
Number of Destination Node	2 - 8
Number of Links per Node	1-5
DQN Layer	2
Activation Function	ReLu
Ratio of Learning Cases	30%, 50%, 70%

In the case of using 50% of learning cases, the same experiments were performed as were done with 70% of learning cases. The experimental results are shown in Fig. 7, and there was no section in which the performance converged. In the case of 30,000 learning episodes, the experiments were carried out again in the increment steps of 9,000 episodes, as shown in Fig. 8. In the experimental results, the case of 75,000 episodes showed the best performance with the performance ratio of 1.21.

Fig. 9 shows the proportion of optimal and suboptimal trees found according to the number of destinations using the value function that has learned the episode 75,000 times. On average, 42% of optimal trees were found, and 80% of suboptimal trees were found on average.

In the case of using 30% of learning cases, the same experiments were performed as were done with 70% of learning cases. The experimental results are shown in Fig. 10, and there was no section in which the performance converged. Furthermore, in the case of learning 30,000, the

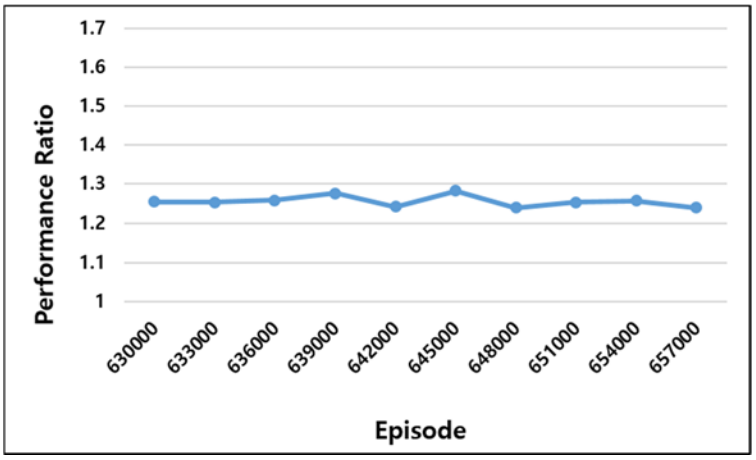


Figure 12: Performance ratio in 30% of learning cases between 630,000 and 657,000 episodes

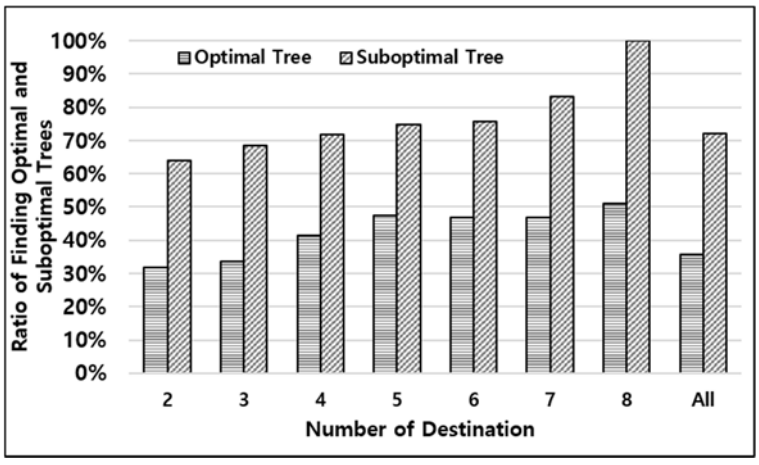


Figure 13: Construction ratio of optimal and suboptimal trees in 30% of learning cases

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

performance decreased instead. In the case of 30% of learning cases, even when additional learning was performed up to 90,000 times, the learning performance was lower than that of the 70% and 50% cases. Fig. 11 shows the results of learning 630,000 times, in increments of 90,000 starting from 90,000. In the results of learning 630,000 times, the average performance ratio was 1.25, indicating the best performance. Since an increasing trend of performance was shown from 540,000 times to 630,000 times, the performance for the learning was investigated again starting from 630,000 and increased in incremental steps of 3,000, as shown in Fig. 12. The performance was highest when the learning was performed 648,000 times, and the performance ratio was 1.239.

Fig. 13 shows the proportion of optimal and suboptimal trees found according to the number of destinations using the value function that has learned the episode 648,000 times. On average, 36% of optimal trees were found, and 72% of suboptimal trees were found on average.

The method used for finding an optimal tree length was the brute force method. All paths of 5,050 cases were searched based on the Breadth-first search (BFS), and after finding an optimal tree length, the performance ratio was derived through the tree that was generated using the proposed method. The 50% learning case showed the best average performance ratio. The performance ratio was 1.21, which showed a maximum improvement in performance of 39.5% and a minimum of 24.2%, compared to that of conventional studies. The result for 50% learning was better than that for 70% learning, and it can be interpreted that in the case of 70%, overfitting occurred with many of the learning cases.

In the 70% case and the 50% case, the frequency of learning that showed the best performance was 21,000 times and 75,000 times, respectively. Moreover, it was confirmed that in the 50% case, learning has to be performed more than three times before it can show better performance than the 70% case. This implies that the learning case proportion and the frequency of learning should be adjusted appropriately according to the network situation in multicast routing.

6. CONCLUSIONS

This study proposed a single multicast tree construction method using DQN, which is a reinforcement learning method among machine learning methods in the SDN environment. The learning environment of this study had much learning data, and the situation is appropriate for application of a DQN because correlation between data is high. Therefore, through a DQN, a policy was learned, and it generated the value function, thereby constructing a multicast tree. Furthermore, after performing the learning according to the proportion of learning cases, the tree was created through the generated model, and the performance ratio was measured in the experiments. In the results, the 50% case showed the best performance when the episode was learned 75,000 times. The average performance ratio was 1.21, which showed an improvement of 24.2% minimum and 39.5% maximum over the conventional TM algorithm.

This paper has also limitations. To resolve the limitations, there is a plan for a future study to analyze the performance change by learning rate. Furthermore, the experiments will be conducted with an increase the number of nodes from the current ten nodes. The plan is to perform the learning using two or more layers in the neural network. We also plan to study a method of setting and applying rewards for various cases, excluding the optimal case, and a method of generating multiple multicast trees instead of a single multicast tree. An additional study will be conducted for a situation in which the topology changes dynamically.

ACKNOWLEDGMENTS:

This work was supported by Kyonggi University Research Grant 2017.

REFRENCES:

[1] Cisco, V. N. I. "Cisco visual networking index: Forecast and trends, 2017–2022," White Paper 1, 2018.

[2] C. A. C. Marcondes and T. P. C. Santos, "CastFlow: Clean-slate multicast approach using in-advance path processing in programmable networks," 2012 IEEE Symposium on Computers & Communications (ISCC), pp. 94-101, 2012.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[3] S. Xu, C. Wu, and Z. Li, "Software defined mobile multicast," Mobile Ad Hoc and Sensor Systems (MASS), pp. 208-216, 2015.

[4] N. Xue, X. Chen, L. Gong, S. Li, D. Hu, and Z. Zhu, "Demonstration of OpenFlow-controlled network orchestration for adaptive SVC video manycast," IEEE Transactions on Multimedia, Vol. 17, pp. 1617-1629, Sep. 2015.

[5] J. R. Jiang and S. Y. Chen, "Constructing multiple Steiner trees for software-defined networking multicast," ACM International Conference Proceeding Series, pp. 1-6, 2016.

[6] M. Sun, et al., "A multiple multicast tree optimization solution based on software defined network," Information and Communication Systems (ICICS), pp. 168-173, 2016.

[7] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C. G. Liu, and L. Wei, "The PIM architecture for wide-area multicast routing," IEEE/ACM transactions on networking, Vol. 2, pp.153-162, 1996.

[8] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet group management protocol (IGMP)/multicast listener discovery (MLD)-based multicast forwarding (IGMP/MLD Proxying)," IETF RFC 4605, 2006.

[9] D. Waitzman, S. E. Deering, and C. Partridge, "Distance vector multicast routing protocol," 1988.

[10] P. Rajvaidya and K. C. Almeroth, "Analysis of routing characteristics in the multicast infrastructure," IEEE INFOCOM 2003, Vol. 2, pp. 1532-1542, Mar. 2003.

[11] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S., Deering, M., Handley, and L. Wei, "Protocol independent multicast-sparse mode (PIM-SM): Protocol specification," 1998.

[12] A. Adams, J. Nicholas, and W. Siadak, "Protocol independent multicast-dense mode (PIM-DM): Protocol specification (revised)," IETF RFC 3973, Jan. 2005.

[13] M. Handley, L. Vicisano, I. Kouvelas, and T. Speakman, "Bidirectional protocol independent multicast (BIDIR-PIM)," 2007.

[14] H. Holbrook and B. Cain, "Source-specific multicast for IP," IETF RFC 4607, Aug. 2006.

[15] N. McKeown, et al, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM 2008, Vol. 38, pp. 69-74, Apr. 2008.

[16] ONF, "Software-Defined Networking: The new norm for networks," ONF White Paper, pp. 7, Apr. 13, 2012.

[17] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," IEEE Communications Magazine, Vol. 51, No. 7, pp. 36-43, July 2013.

[18] J. Moy, "Multicast routing extensions for OSPF," Communications of the ACM, Vol. 37, No. 8, pp. 61-67, 1994.

[19] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT)," ACM SIGCOMM 1993, Vol. 23, No. 4, pp. 85-95, Oct. 1993.

[20] R. Novak, J. Rugelj, and G. Kandus, "Steiner tree based distributed multicast routing in networks", Steiner Trees in industry, Springer, pp. 327-351, 2001.

[21] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," Math. Japonica, Vol. 24, No. 6, pp. 573-577, 1980.

[22] Markov Decision Process, https://en.wikipedia.org/wiki/Markov_decision_process

[23] Q-learning Variants, https://en.wikipedia.org/wiki/Q-learning#Variants

[24] V. Mnih, K., Kavukcuoglu, D., Silver, A. A. Rusu, J. Veness, M. G. Bellemare, and S. Petersen, "Human-level control through deep reinforcement learning," Nature, 518.7540: 529. 2015.