Reinforcement Learning Based Dynamic Resource Migration for Virtual Networks

Takaya Miyazawa, Ved P. Kafle, and Hiroaki Harai
National Institute of Information and Communications Technology (NICT),
4-2-1 Nukui-Kitamachi, Koganei-shi, Tokyo, 184-8795 Japan
E-mail: {takaya, kafle, harai}@nict.go.jp

***Abstract*— Network virtualization techniques enable network operators to implement and provide multiple virtual networks (VNs) on a common substrate network infrastructure. The network resources should be able to be dynamically reallocated among VNs or migrated from a place to another one in various situations, such as to construct a new urgent VN in case of occurrence of some urgent contingency, or to satisfy quality of service (QoS) requirements of non-urgent VNs in case of time-varying network traffic conditions as much as possible. In this paper, we propose methods to automatically and dynamically select and migrate resources of non-urgent VNs. In particular, our proposal applies reinforcement learning for the selection of resources from alternate places to satisfy the QoS requirements. We evaluate the performances in terms of the satisfaction level of QoS requirement with various frequency of network traffic variation in a given duration and learning parameter configurations. Simulation results show that the proposed dynamic resource migration method can increase the number of times that non-urgent VNs' QoS requirements are satisfied in comparison with a static resource assignment method. Moreover, we show that, depending on traffic variation and parameter configuration, applying reinforcement learning can increase the number of times the QoS requirement is satisfied compared to the dynamic method with completely random resource selection.**

Keywords—Virtualization; Resource Migration; Reinforcement Learning; Quality of Service.

I. INTRODUCTION

Recently, many researchers have been investigating techniques for the virtualization of networks, servers, and network functions [1]-[4]. One of the advantages of virtualization is that it realizes isolation. For example, virtual networks (VNs) configured through network virtualization on the top of an underlying substrate infrastructure can be isolated from each other by allocating and managing network resources, network functions, or flow spaces separately. In [1], FlowVisor is proposed as a network virtualization technique. It can be implemented on OpenFlow-based packet-switched networks, and realizes network virtualization by allocating separate flow spaces at multiple layers (e.g. by using source/destination MAC address, source/destination IP address, VLAN ID, and source/destination TCP/UDP port) to each VN. In the future networks, virtual network operators (VNOs) will appear as entities for network service provisioning, and network virtualization enables multiple VNOs to provide and manage individual resources, functions or flow spaces on separate VNs constructed on a common physical substrate network. For example, as for mobile networks, mobile VNOs are already providing virtual network services by leasing physical lines from other carriers.

Current network construction mainly depends on manual operations for network resource allocation, identifiers (IP address, VLAN ID, etc.) allocation, parameter configurations, management, and so on. Hence, automation of such operations will be essential to shorten network configuration time, avoid human errors, or deal with shortage of human resources. Moreover, data generating devices (e.g. PCs, mobile/smart phones, vehicles, sensors, and robots) and quality-of-service (QoS) requirements (e.g. packet loss rate, latency and jitter) will be increasingly diverse due to rapid prevalence of Internet-of-Things (IoT) applications in the future. Especially in view of using mobile terminals, VNs should be constructed by recognizing both time-varying network environments (e.g. network traffic and failure status) and diversified service requirements (i.e. service levels specified by application service providers). In other words, VNOs will be required to automatically and dynamically select virtualized resources suited for each VN by recognizing networking environments and service requirements. Here, let us assume that some urgent contingency (e.g. due to traffic accidents, natural disasters, security enhancement against massive crimes, or some short-notice social events) occurs, and a VN for the contingency needs to be constructed. In such a case, the infrastructure provider (InP) should ensure requisite resources for the VN, and already-allocated resources to non-urgent VNs may have to be reallocated to the urgent VN so that the quality of service (QoS) requirement of the urgent VN can be guaranteed. Meanwhile, as mentioned above, the network traffic tends to be time-varying, especially in networks of the future IoT applications. In such network environments, non-urgent VNs also need to automatically and dynamically perform migration of resources for satisfying their QoS requirements to a maximum extent when their traffic volume fluctuates.

The authors in [2] propose migration of network functions (intrusion detection system, NAT, load balancer, caching proxy, traffic monitoring, etc.) from an instance to another; along with a migration process, switches in relevant nodes (on both old and new routes) are controlled by software-defined networking (i.e. OpenFlow). The authors in [3] construct a combined server virtualization environment using Host Hypervisors and network virtualization environment using a Network Hypervisor, and create and migrate logical forwarding data paths between virtual machines (VMs). The authors in [4] propose a framework for mapping a service function chain (SFC) onto physical network infrastructure of nodes and links.

An SFC is created by connecting virtual network functions (VNFs) by logical links, and the proposed method has a resource migration function to protect the SFC from VNF failures. The authors in [5] propose inter-domain VM migration, in which network controllers and a global orchestrator select and change the route by means of OpenFlow when they receive a VM migration request from Cloud OS. However, these exiting researches lack sufficient discussion on how to dynamically decide and update about where to migrate each VN's resource in consideration of time-varying network traffic and QoS requirement of each VN. Especially, there is no prior work on application of machine learning (ML) to satisfy QoS requirements in time-varying core network environments. The authors in [6] propose to apply a ML technique to resource migration; however, they only target at cloud & VMs environment, and do not touch the issue of resource migration in a large-scale network consisting of edge/core networks and datacenters, where multiple VNs requiring diverse QoS levels are constructed on the common substrate network.

In this paper, we propose methods to dynamically select and migrate virtual resources of (non-urgent) VNs. The most straightforward way to realizing dynamic resource migration is to randomly migrate a VN's resources from a place to another one to satisfy its QoS requirement. However, it is not an optimal solution. Therefore, in our methods, we propose to apply reinforcement learning for the selection of alternate resources to satisfy the QoS requirement. The reinforcement learning [7][8] is classified as a ML technique, and has been applied to various types of network control [6][9][10]. We evaluate the performance in terms of the satisfaction level of QoS requirement with various frequency of network traffic variation in a given duration and learning parameter configurations. By computer simulations, we demonstrate that the dynamic resource migration methods can increase the number of times that a non-urgent VN's QoS requirement is satisfied in comparison with static resource utilization, and moreover, applying the reinforcement learning can further increase the number of times that the QoS requirement is satisfied compared to the dynamic method with completely random selection.

II. VIRTUAL NETWORK CONSTRUCTION

Figure 1 illustrates (a) structure of physical and virtual networks and (b) mapping of virtual resources onto physical resources. In the physical networks, edge networks and data centers are connected to a large-scale core network. The edge networks are composed of equipment collecting/processing data from various IoT terminals such as PCs, mobile/smart phones, vehicles, sensors, and robots. In this work, we assume that the whole physical network is managed by a single InP, i.e. a single domain environment. VNs are constructed over the whole physical network consisting of edge/core networks and data centers, and "resources" include computational (CPU, memory, and storage) as well as network (link bandwidth and buffer space in nodes) resources. Each VNO recognizes the service level requirements of an application (which we call service requirements) and/or social/environment conditions (i.e. urgent contingency, network traffic, etc.), and is in charge of VN construction including both creation of logical topology and selection of network and computational resources. The

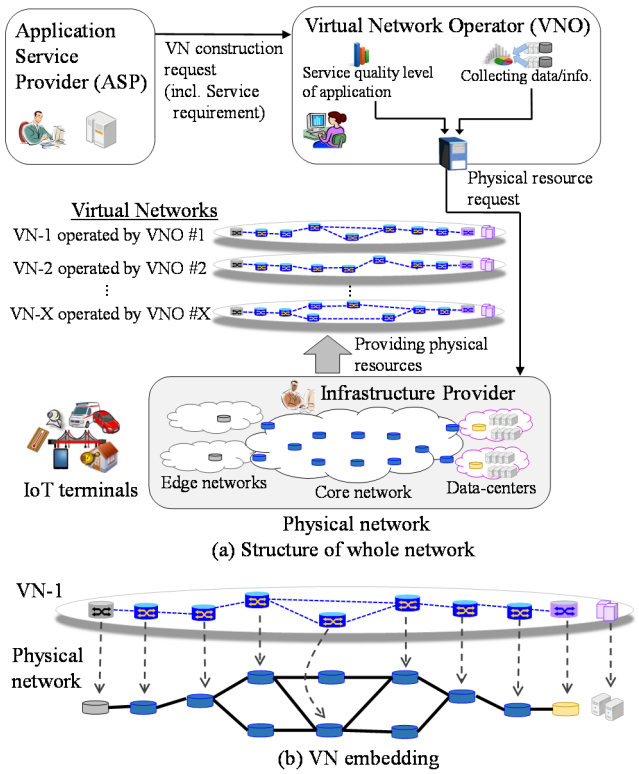

Fig. 1. Images of (a) structure of whole network and (b) VN embedding.

VNO then requests the InP to lease physical resources. The InP performs mapping of virtual resources (nodes and links) onto physical resources in the substrate networks. This mapping process is often referred to as *VN embedding* [11]. Based on the requests from VNOs, the InP leases physical resources to them (e.g. registers or updates each mapping information) for VN construction. VNs' virtual resources may be migrated from a set of physical resources to another set of physical resources for satisfying their QoS requirements. Here, a set of physical

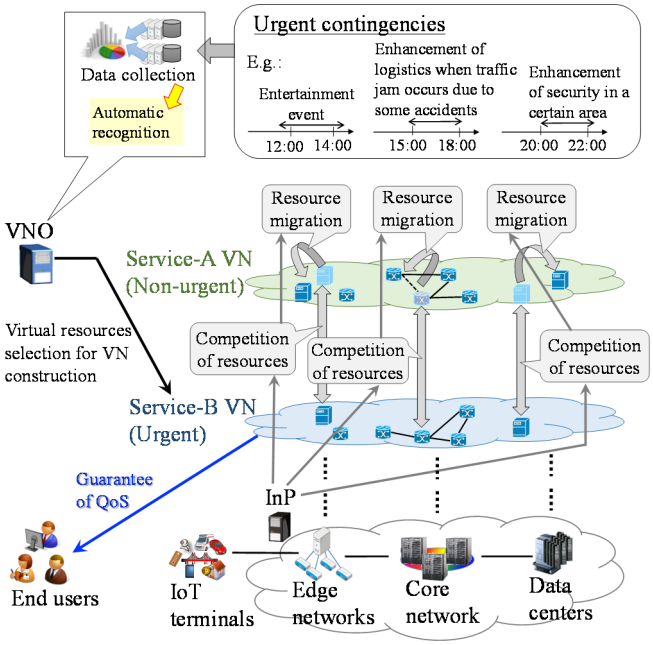

Fig. 2. An image of migration of non-urgent VN's resources.

resources consists of resources in the relevant edge network, core network, and data center. Different VNs may have different logical topologies, as well as various network and computational resources.

Figure 2 illustrates migration of non-urgent VN's resources when an urgent contingency occurs. The VNO recognizes the urgent contingency by analyzing social/environmental data regularly collected by them. In order to construct a VN with high QoS, the VNO selects virtual resources over edge/core networks and data centers, and requests the InP to lease the physical resources. The VN with a high QoS requirement is preferentially allocated with optimal resources. Then, if multiple VNs happen to compete for resources, the resources being used by non-urgent VNs are reallocated to the urgent VN. In this way, the high QoS requirement of urgent VN is always met. Meanwhile, the networks should also be able to maintain the QoS of resource migrated non-urgent VNs to an optimal level.

III. ML-Based Autonomic Resource Control

In order to achieve automation of various operations and handle increasingly diverse IoT devices/applications and QoS requirements, we apply ML techniques in VN construction. Especially for resource control, we propound machine learning-based autonomic resource control (MLARC). Currently, we strive to apply ML to (i) selecting virtual resources in the process of initial VN construction according to a certain VN request [12] and (ii) deciding where to migrate each (non-urgent) VN's resources dynamically in consideration of network traffic situation and QoS requirements.

Figure 3 shows a control process flow diagram of MLARC. Inside the VNO, "Output" is determined by a virtual resource selection process for the initial VN construction on the basis of information of "Input". We have applied a ML technique (i.e. support vector machine tool) to the virtual resource selection process [12]. Here, the input information consists of QoS requirement levels and network parameters. For example, the input can include service requirements on link bandwidth, packet loss rate, delay, and jitter. It can also include network traffic load, failure status, geographical position from where a user receives data, and so on. The output information includes logical topology, network resources such as link bandwidth and buffer space in network nodes, and computational resources such as CPU, memory, and storage space of computer equipment (e.g. server and controller). The VNO sends a request for physical resources to the InP on the basis of the output. The InP executes the VN embedding to map virtualized resources onto physical resources in the substrate networks, and leases physical resources to VNOs. A VN for urgent contingency is preferentially allocated an optimal amount of physical resources. As a consequence, if there are not enough physical resources, non-urgent VNs' resources should be migrated to another set of physical resources by the resource migration process. In this paper, we mainly focus on this resource migration. Especially, we apply reinforcement learning to this process. After the decision about resource migration is made, the InP executes the VN embedding, and then, reallocates physical resources to each relevant VN. After the VN construction is completed, the VNO monitors performance (e.g. packet loss rate and delay) of each VN and

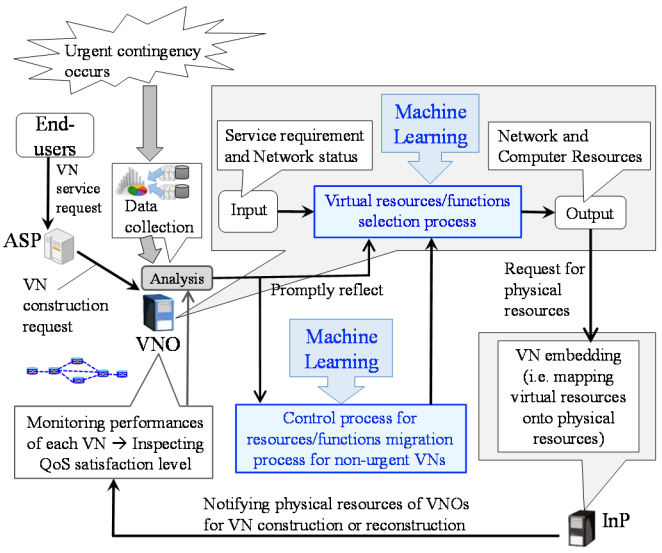

Fig. 3. Control process flow diagram of MLARC.

inspects QoS satisfaction levels. The VNO can promptly reflect the monitoring results in the virtual resource selection process and/or resource migration process.

IV. Proposed Resource Migration Scheme

As mentioned in Section I, the most straightforward way to realizing dynamic resource migration is to randomly select where to migrate a VN's resources when the QoS requirement is not satisfied. Note that, while the QoS requirement is satisfied, no resource migration takes place. The dynamic method is expected to make it easier to satisfy QoS requirement of each (non-urgent) VN. However, it is possible that the dynamic method with completely random selection may experience unfortunate cases where migrated resources still cannot satisfy the QoS requirement. Therefore, we propose to utilize the reinforcement learning to selection of alternate resources in the case that the QoS requirement is not satisfied. Reinforcement learning is one of ML techniques [7][8]. The key parameters of the reinforcement learning process are States, Actions, and Rewards. Figure 4 shows a conceptual diagram of reinforcement learning. Initially, the state starts at the source state. The learning agent perceives the current state of the network environment, and selects and executes one of possible actions by means of an action selection method. Generally, learning agent and environment are regarded as a controller and a controlled object, respectively. The agent function is installed inside each VNO, and executes control processes for resource migration. The state is shifted from the current state to the next state by executing an action. Through iterative trial and error, a reward is given to every action or to the action to

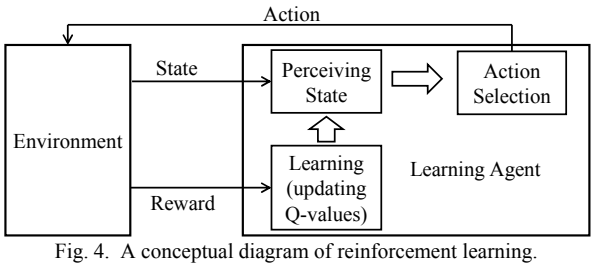

Fig. 4. A conceptual diagram of reinforcement learning.

reach the goal (destination state), and the learning agent updates Q-values which express how much worthy the actions are. The learning agent iteratively learns actions to optimize (e.g. maximize) the overall reward between source and destination states. In this paper, for simplicity, we assume that a single agent system is installed in the VNO although multiple agents system will be effective at achieving higher scalability. The reinforcement learning is a systematic trial and error approach, and one of major learning algorithms is Q-learning [6]-[9]. We use the Q-learning in this paper.

Our proposed method defines States, Actions, and Rewards as in the typical process of Q-learning. Actions are stochastically selected through iterative learning processes, and resource migration is executed in accordance with the selected set of actions. Depending on parameter configuration, an action with higher Q-value (i.e. high-value action) tends to be selected, and it results in improvement of performances (i.e. better satisfaction of QoS requirements). Meanwhile, note that "partly" random selection is preferable in order to avoid the situation that resources of many VNs are migrated to the same set of physical resources and congestion may occur there. In the reinforcement learning, there is an action selection method which is stochastically executed on the basis of one of the following two logics: (1) selecting an action with higher Q-value, or (2) selecting an action randomly. It can be formulated as described below.

We define States as $s_i \in S = \{SRC, E_1, E_2, \ldots, E_J, C_1, C_2, \ldots, C_K, D_1, D_2, \ldots, D_L, DST\}$, which varies through a learning process. Here, *SRC* is the source node, E_x ($x = 1, 2, \ldots, J$) are edge nodes, C_x ($x = 1, 2, \ldots, K$) are core nodes, D_x ($x = 1, 2, \ldots, L$) are data center nodes, and *DST* is the destination node. The values of *J*, *K*, and *L* represent the numbers of nodes in the edge network, core network, and data center, respectively, in a single domain network. The variable *i* corresponds to the hop number starting from SRC, e.g. for $i = 0$, $s_i = SRC$. We also define Actions as the decision of selection of a link connecting the current node to the next node, which are expressed as $a_i \in A(s) = \{A_1(s), A_2(s), \ldots\}$, where $A_x(s)$ means the action of selection of link index x at a state $s \in S$. By executing action a_i, the state transits from s_i to s_{i+1}. Additionally, we define Rewards as QoS satisfaction levels of each VN. Reward r_i is the reward obtained when action a_{i-1} is executed at state s_{i-1}. A reward is given to an action in accordance with the QoS satisfaction level. The higher the QoS satisfaction level is, the higher the reward is. In this paper, for simplicity, we focus only on packet loss rate (PLR) as the QoS parameter. Reward r_i is expressed as

$$r_i = \begin{cases} \{1 - p_M / P_R\} \times \beta & \text{if } p_M < P_R, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where p_M is the measured (monitored) PLR between *SRC* and *DST*, P_R is the requirement on PLR for the VN, and β is a positive integer to adjust reward values. In this work, a reward is obtained only when the state arrives at *DST*. Namely, when $s_{i+1} = DST$, reward r_{i+1} is given to action a_i. As we can see in Eq. (1), a lower PLR results in the higher value of reward.

We denote Q-value in the case that action a_i is selected at state s_i by $Q(s_i, a_i)$. All Q-values are initially set to 0. In every learning process, when state s_i is shifted to state s_{i+1} by performing action a_i, the Q-values are updated by use of the Q-learning rule expressed as

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha \times [\, r_{i+1} + \gamma \times \max_{a \in A(s)} Q(s_{i+1}, a) - Q(s_i, a_i) \,] \quad (2)$$

where α is learning rate ($0 \le \alpha \le 1$, which is usually set as around 0.1), and γ is discount factor ($0 \le \gamma \le 1$, which is usually set as a value in the range from 0.90 to 0.99).

The model and formulation in our proposed method have an advantage that it can give a reward every time the state reaches a transit node as well as DST. Note that we could also use another model and formulation (e.g. a state is a current end-to-end route, an action is moving the VN from one state to a new one, and a reward is given only at DST).

For an action selection, we propose to apply ε–greedy method due to its simplicity and practicality [8]. Action selection is performed only when the QoS requirement is not satisfied and resource migration is needed. Our method selects an action that gives the maximum value of $Q(s_i, a_i)$ with the probability of $(1 - \varepsilon)$, and randomly selects an action with the probability of ε. The value of ε can be dynamically changed depending on the situation. The learning agent selects an action at each state independent of actions selection at other states between *SRC* and *DST*. Next to the state *DST*, the state goes back to *SRC*. One cycle from *SRC* to *DST* is called a learning process. By the iterative learning processes, the reward is reflected in Q-values at previous states/actions in series. The learning agent performs the prescribed number of learning processes, and finally decides resources for the VN and resource migration is executed in accordance with the set of actions that have been finally decided. $Q(s_i, a_i)$ can have only one value at a time, and thus, each Q-value is dynamically updated through the iterative learning processes. An advantage of reinforcement learning is that the Q-values associated with a route where the QoS requirement has been continuously satisfied in the past get higher and higher, and the route tends to be selected with a higher probability. We assume network environments that each VNO cannot get complete information about other VNOs' decisions on resources selection; thus, each agent cannot find the optimal solution (e.g. by linear programming). Even if the complete information can be obtained, the time needed to find the solution drastically increases as the number of VNs is larger and especially in unpredictable network environments. Our proposed method is not affected by the number of VNs because it selects a route for a VN independently of routes selected for other VNs. Moreover, it performs trial-and-error for dynamic resource migration in such environments.

V. EVALUATIONS

We evaluated our proposed scheme to show the effectiveness of dynamic resource migration and application of reinforcement learning. As a first step, we performed computer simulations with some simple assumptions as explained below: We assume the network topology depicted in Figure 5. In the edge network, there are one source node (*SRC*) and two edge nodes (E_1 and E_2). In the core network, there are five core nodes ($C_1, C_2, \ldots, C_5$). In the data center, there are three data center nodes (D_1, D_2, and D_3) and a destination node (*DST*). It is assumed that an urgent contingency occurs and its VN occupies all physical resources on the route of $SRC - E_1 - C_2 -$

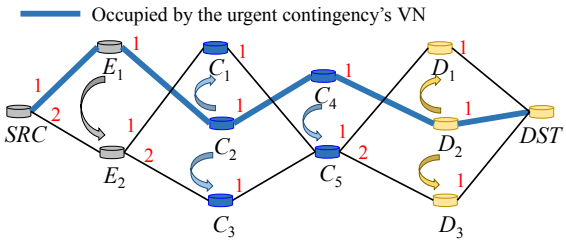

Fig. 5. Network topology for simulation.

$C_4 - D_2 - DST$ which we call as the urgent route. We focus on an existing non-urgent VN with the QoS requirement of PLR $\leq 10^{-3}$, and its virtual resources are migrated from the physical resources on the urgent route to the other route which does not get through the urgent route. We assume that the non-urgent VN is constructed on bandwidth-shared (e.g. packet-switched) type of networks. This means that its physical resources are shared with other network traffic including best-effort background traffic although it might be better if some nodes were able to provide dedicated resources to the VN in order to reliably guarantee the QoS. Additionally, other background traffic is coming into C_1, C_3, and C_5 from outside networks. Thus, it is too difficult to predict the amount of network traffic. In Fig. 5, the numbers ("1" and "2") in red font are link interface IDs, which also correspond to the action numbers; for example, when $s_i = E_2$ and $s_{i+1} = C_3$, $a_i = 2$. Note that, in this simple network topology, the migrated non-urgent VN has to get through E_2 and C_5 between SRC and DST inevitably.

Table I shows the possible combinations of state s_i, action a_i and state s_{i+1}. Note that the 1st, 7th, 9th, 13th lines cannot be selected because the urgent contingency's VN occupies the relevant physical resources. At states E_2 and C_5, there are two actions A_1(s) and A_2(s) from each state. This means that, in this paper, we assume that there are four possible end-to-end resources between *SRC* and *DST*. For the sake of convenience, we call the route of $SRC - E_2 - C_1 - C_5 - D_1 - DST$ as Route-1, $SRC - E_2 - C_1 - C_5 - D_3 - DST$ as Route-2, $SRC - E_2 - C_3 - C_5 - D_1 - DST$ as Route-3, and $SRC - E_2 - C_3 - C_5 - D_3 - DST$ as Route-4.

We obtained and compared performances of three resource migration methods: (A) Static resource migration method, (B) Dynamic resource migration with completely random selection method, and (C) Dynamic resource migration with reinforcement learning method. When an urgent contingency occurs, the learning agent selects a certain route for the non-urgent VN, which is Route-1 in this simulation. After that, in method (A), the route does not change during the lifetime of the VN. In method (B), a route is decided and changed by completely random selection when the QoS requirement is not satisfied. While the QoS requirement is satisfied, resource migration is not executed. In method (C), a route is decided and changed in accordance with the result of the iterative Q-learning processes formulated in Eq. (2). While the QoS requirement is satisfied, resource migration is not executed as in the case of method (B). We set the values of β, α, and γ as 10, 0.1, and 0.95, respectively.

Table I. Possible combinations of state s_i, action a_i, and state s_{i+1}.

	State s_i	Action a_i	State s_{i+1}	Selectable or Not for non-urgent VNs
$i = 0$	*SRC*	1	E_1	Not
	SRC	2	E_2	Selectable
$i = 1$	E_1	1	C_2	Selectable
	E_2	1	C_1	Selectable
	E_2	2	C_3	Selectable
$i = 2$	C_1	1	C_5	Selectable
	C_2	1	C_4	Not
	C_3	1	C_5	Selectable
$i = 3$	C_4	1	D_2	Not
	C_5	1	D_1	Selectable
	C_5	2	D_3	Selectable
$i = 4$	D_1	1	*DST*	Selectable
	D_2	1	*DST*	Not
	D_3	1	*DST*	Selectable
$i = 5$	*DST*	-	-	-

We define a unit time slot T_U as the time interval required for both route selection and resource migration for a VN, and t_f as the time interval in which the amount of network traffic (and thus PLR) changes. We also define c_f as the ratio of t_f to T_U, which we call the network traffic variation interval; the unit is slot (i.e. $c_f = t_f / T_U$). Note that fluctuation of network traffic causes fluctuation of PLRs in each link. We assume $t_f \geq T_U$ and c_f is positive integer for simplicity. The follow-up of reinforcement learning should be done within the time of much less than T_U. We classify links into four kinds of transparent links: $E_2 - C_1 - C_5$, $E_2 - C_3 - C_5$, $C_5 - D_1 - DST$, and $C_5 - D_3 - DST$. The transparent link means that there is no branching along the way between both ends of the link. We generate a value of PLR on each transparent link by a random number generator in the range from 10^{-5} to 10^{-1} in each time slot in which PLRs change (by using different values of random seeds). Thus, we can obtain the value of PLR on each route by simply adding the PLRs on two relevant transparent links [13]. In method (C), we assume that the learning agent in VNO can obtain the PLR of any route.

In this evaluation, we set the number of time slots as 300, which corresponds to the execution time of this simulation. In other words, the maximum number of resource migration steps

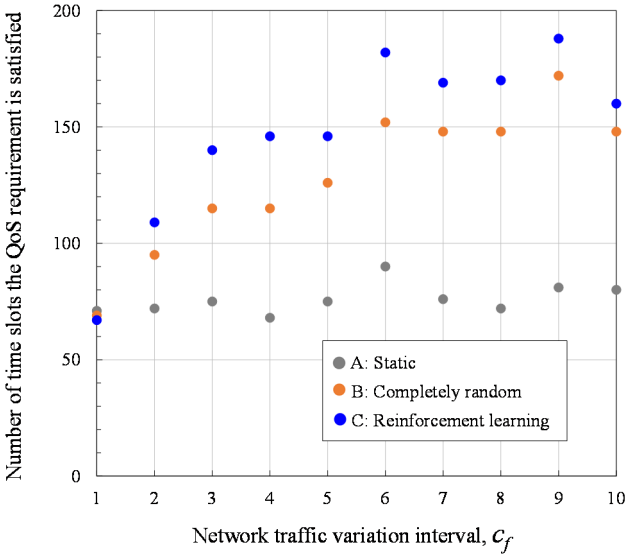

Fig. 6. The number of satisfied slots versus c_f, where $n_i = 10$ and $\varepsilon = 0.2$.

is 300, and also the upper limit of time slots the QoS requirement is satisfied is 300. Figure 6 shows the number of time slots the QoS requirement is satisfied (which is called "the number of satisfied slots" below) versus the network traffic variation interval c_f. Here, we have set values of the number of iterations of learning in one process of route selection, n_i, as 10 and ε as 0.2. We can see that the dynamic methods (B) and (C) can drastically increase the number of satisfied slots compared to the static method (A) in the majority of cases. This is because the dynamic methods execute resource migration when the PLR exceeds 10^{-3}, while the static method does not execute it even if the requirement on PLR is not satisfied. Note that, when $c_f = 1$, the performance of static method is almost the same in comparison with dynamic methods because of the fact explained below: It is probable that, just after the network traffic changes, the PLR of route selected in the static method (i.e. Route-1) does not exceed 10^{-3} while the PLR of route selected in method (B) or (C) exceeds 10^{-3}. Especially, when the network traffic variation interval is very short, it might have adverse effects on the dynamic methods depending on the situation such as traffic fluctuation patterns and parameters configuration. In fact, we can see that, as the value of c_f becomes higher, the improvement effects of dynamic methods increase in comparison with the static method. Meanwhile, method (C) can achieve higher number of satisfied slots compared to method (B) because method (C) tends to select a set of actions with higher Q-values through iterative learning processes. When a route satisfies the QoS requirement of the VN, the relevant Q-values increase by the reward formulated in Eq. (1) in Section IV.

Figure 7 shows the number of satisfied slots versus action selection parameter in reinforcement learning ε. We set the value of n_i as 10 for method (C). The dots indicate the performances of method (C), and the solid lines indicate those of method (B). In comparison with the performances of method (B), method (C) can improve the performances in the majority of cases in this parameter configuration (i.e. n_i), but method (B) is better only when $c_f = 1$ and the value of ε is not 0.1. In this simulation, we obtain a value of PLR on each link by a random number generator as already mentioned. This means that, when the network traffic (reflecting in PLRs) changes, method (C) may increase the number of time slots the QoS requirement is not satisfied depending on previous action selection results even though a set of actions with higher Q-values was selected. In other words, when the network traffic variation interval is very short, we may not be able to receive the benefit from reinforcement learning because the traffic fluctuation is too fast and it is probable that a result of resource migration causes QoS unsatisfaction. Meanwhile, the performances change when we change the value of ε. It is effective to dynamically set the value of ε in consideration of network situation.

Figure 8 shows the number of satisfied slots versus the number of iterations of learning in one process of route selection n_i. We set the value of c_f as 5. As shown in Fig. 6, when $c_f = 5$, the number of satisfied slots in method (B) is 126. In comparison with method (B), method (C) achieves better performances at all times when the value of n_i is equal to or larger than 2. Besides, the numbers of satisfied slots in method (C) become larger as the value of n_i increases, and converge more or less when the value of n_i exceeds 9. As the value of n_i is smaller (e.g. 1 or 2), the number of iterations of updating Q-values decreases, and thus, the reward cannot be reflected in the Q-values related to a route where the QoS requirement is satisfied. If enough number of iterations are executed, the reward can be reflected in all Q-values related to the route. Thus, it results in convergence of performances. Meanwhile, as method (C) adopts the ε-greedy method for an action selection and an action is selected randomly with the probability of ε, it results in variation of performances along with change of n_i.

VI. CONCLUSION

We proposed the methods to dynamically select/migrate resources of VNs when some urgent contingency occurs and a highest-priority VN needs to be constructed. Our proposal applies the reinforcement learning to selection of alternate resources to satisfy non-urgent VNs' QoS requirements. We demonstrated that the proposed method can increase the number of times that a non-urgent VN's QoS requirement (on packet loss rate limit) is satisfied in comparison with both the static and dynamic methods with completely random selection.

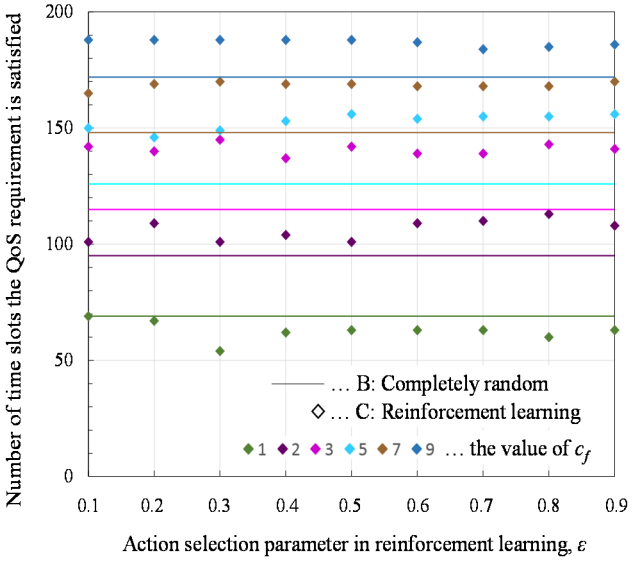

Fig. 7. The number of satisfied slots versus the value of ε, where $n_i = 10$.

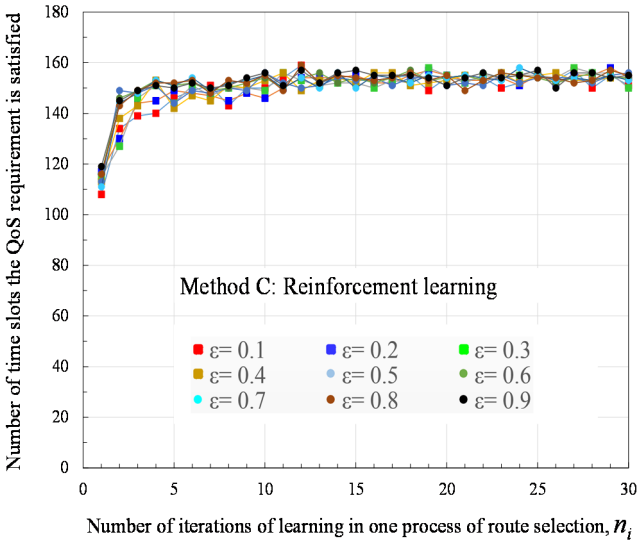

Fig. 8. The number of satisfied slots versus the value of n_i where $c_f = 5$.

REFERENCES

[1] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer," *OpenFlow Switch Consortium, Technical Report*, Oct. 2009.

[2] A. G. Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," *Proc. of ACM SIGCOMM '14*, Chicago, IL, USA, pp.163-174, Aug. 2014.

[3] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, N. Gude, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S. H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, "Network Virtualization in Multi-tenant Datacenters," *Proc. of 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14)*, Seattle, WA, USA, pp.203-216, April. 2014.

[4] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing Reliabilty with Enhanced Protection in NFV," *Proc. of ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox'15)*, London, UK, pp.13-18, Aug. 2015.

[5] J. Liu, Y. Li, and D. Jin, "SDN-based Live VM Migration Across Datacenters," *Proc. of ACM SIGCOMM '14*, Chicago, IL, USA, pp.583-584, Aug. 2014.

[6] M. Duggan, J. Duggan, E. Howley, E. Barrett, "An Autonomous Network Aware VM Migration Strategy in Cloud Data Centres," *Proc. of IEEE International Conference on Cloud and Autonomic Computing (ICCAC 2016)*, Augsburg, Germany, pp.24-32, Sep. 2016.

[7] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol.4, pp.237-285, Jan. 1996.

[8] Y. Mohan and Ponnambalam S. G., "Q-learning Policies for a Single Agent Foraging Tasks," *Proc. of 7th International Symposium on Mechatronics and its Applications (ISMA '10)*, Sharjah, UAE, pp.1-6, Apr. 2010.

[9] I. Koyanagi, T. Tachibana, and K. Sugimoto, "A Reinforcement Learning-Based Lightpath Establishment for Service Differentiation in All-Optical WDM Networks," *Proc. of IEEE GLOBECOM 2009*, Honolulu, HI, USA, pp.1-6, Nov.-Dec. 2009.

[10] J. V. D. Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. D. Turck, "A Learning-Based Algorithm for Improved Bandwidth-Awareness of Adaptive Streaming Clients," *Proc. of IEEE/IFIP International Symposium on Integrated Network Management (IM 2015)*, Ottawa, ON, Canada, pp. 131-138, May 2015.

[11] F. Esposito and I. Matta, "A Decomposition-Based Architecture for Distributed Virtual Network Embedding," *Proc. of ACM SIGCOMM Workshop on Distributed Cloud Computing (DCC'14)*, Chicago, IL, USA, pp.53-58, Aug. 2014.

[12] T. Miyazawa and H. Harai, "Supervised Learning Based Automatic Adaptation of Virtualized Resource Selection Policy," *Proc. of 17th International Network Strategy and Planning Symposium (IEEE Networks 2016)*, Montreal, QC, Canada, pp.170-175, Sep. 2016.

[13] ITU-T Recommendation Y.1541 (12/11) "Network performance objectives for IP-based services". https://www.itu.int/rec/T-REC-Y.1541/en