社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

遅延プロファイルを用いたゆらぎ原理にもとづくSDI仮想化基盤制御手法の提案と評価

井上　昴輝†　荒川　伸一†　今井　悟史††
片桐　徹††　関屋　元義††　村田　正幸†

† 大阪大学 情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
†† （株）富士通研究所　〒 211–8588 神奈川県川崎市中原区上小田中 4–1–1
E-mail: †{k-inoue,arakawa,murata}@ist.osaka-u.ac.jp, ††{imai.satoshi,torukata,sekiya.motoyosh}@jp.fujitsu.com

あらまし　SDI (Software Defined Infrastructure) 環境において通信需要とサービス需要の変動に対し仮想ネットワークに物理的な資源割り当てを行う VNE (仮想網埋め込み) 問題がある。本稿では、大規模複雑かつ不確実な SDI 環境のために少ない情報量で動作する VNE 手法として、制御目標の汎用性を備え、かつ、多重スライス構成に対応可能な VNE 手法として、ゆらぎ原理にもとづく VNE 手法を提案する。計算機シミュレーションによる評価により、提案手法における仮想網の移行回数が、既存の発見的手法と比較して、約 45%削減されることを示す。
キーワード　仮想網埋め込み, SDI, ゆらぎにもとづく仮想網制御, SDN, NFV, 仮想ノードマッピング, 遅延プロファイル

Yuragi-based Approach with Delay Profile for Virtual Network Embedding in Software Defined Infrastructure

Koki INOUE†, Shin'ichi ARAKAWA†, Satoshi IMAI††,
Toru KATAGIRI††, Motoyoshi SEKIYA††, and Masayuki MURATA†

† Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita, Osaka 565–0871, Japan
†† Fujitsu Laboratories Ltd.
Kamikodanaka 4–1–1, Kawasaki-shi, Kanagawa,211–8588 Japan
E-mail: †{k-inoue,arakawa,murata}@ist.osaka-u.ac.jp, ††{imai.satoshi,torukata,sekiya.motoyosh}@jp.fujitsu.com

Abstract　SDI (Software Defined Infrastructure) provides virtualized infrastructures to customers by slicing computing resources and network resources. One of the important problems for deploying SDI framework is to control the assignment of physical resources to a virtual network against changes of traffic demand and service demand. For this problem, VNE (Virtual Network Embedding) problem that maps a virtual network to physical resources has been addressed, but a centralized calculation was assumed. It is difficult to adopt the centralized approaches as a size of infrastructure increases and a number of VN requests increases. This is because the identification of current situation becomes more complicated. In this paper, we present a VNE method that works with a little information for the large, complicated, and uncertain SDI frameworks. To achieve this, the proposed method applies biological "Yuragi" principle. Yuragi, which is a Japanese word, represents a small perturbation to the system. Yuragi is a mechanism of adaptability of organisms and is often expressed as attractor selection model. This paper develops Yuragi-based VNE method that deals with node attribute and has a generality of a performance objective and runs in multi-slice environments. Simulation results show that the Yuragi-based method decreases VN migrations by about 45% than a heuristic method to adapt the fluctuations in resource requirements.
Key words　Virtual Network Embedding, Software Defined Infrastructure, Yuragi-based VNT control, Software Defined Networks, Network Function Virtualization, Virtual Node Mapping, Delay Profile

1 Introduction

SDI provides virtualized infrastructures to customers by slicing computing resources and network resources. In recent years, SDN (Software Defined Networking) and NFV (Network Function Virtualization) technologies have been expected to replace the conventional network management systems, and standardization of SDN/NFV technologies is being promoted [1, 2]. Although SDN/NFV technologies and their standardization are important for deploying SDI, another important problem is to control the assignment of physical resources to a virtual network against changes of traffic demand and service demand. For this problem, VNE (Virtual Network Embedding) problem has been addressed [3, 4]. The VNE problem is a placement problem to allocate virtual resources to the physical network with the optimization of some performance objectives.

The VNE problem is divided into two sub-problems; virtual node mapping and virtual link mapping. In [3–6], a centralized calculation was assumed to solve virtual node mapping and virtual link mapping. That is, a centralized component gathers traffic information and resource utilization of each VN, and identify the current situation of networks. Then, the component solves the optimization problem that optimizes some metrics such as maximizing revenue or minimizing resource utilization. However, when the scale of network size gets larger and the number of multiplexed VNs increases, the identification of current situation gets complicated with enormous network information. This will lead to the occupation of link bandwidth, an increase of delay, and a bottleneck of network scalability [2]. Therefore, it is difficult to adopt the centralized approaches as a size of infrastructure increases and a number of VN requests increases. Moreover, the environments surrounding the Internet today are continuously changing, thus, the adaptive control of VNE is required to handle uncertain changes of environments.

In this paper, we present a VNE method which works with a little information for the large, complicated, and uncertain SDI frameworks. To achieve this, the proposed method applies biological "Yuragi" principle. Yuragi, which is a Japanese word, represents a small perturbation, both externally and internally generated, to the system. Yuragi is a mechanism of adaptability of organisms and is expressed as attractor selection model. Our research group has developed virtual network control based on attractor selection for optical networks. Our results showed that our control mechanism shows the high adaptability to environmental fluctuations with restricted information [7]. Unlike the virtual network on optical networks, virtual network on SDI frameworks has to consider various matters such as node attribute, computing performance on servers, and VN multiplexing. Therefore, this paper develops Yuragi-based VNE method that deals with node attribute and has generality of a performance objective and runs in multi-slice environments.

The rest of this paper is organized as follows. In Sec. 2, a service model in SDI frameworks is introduced and related works on VNE are referred. The method based on Yuragi principle is proposed in Sec. 3, and the results of performance evaluation are shown in Sec. 4. Finally, the conclusion of this paper and future work are presented in Sec. 5.

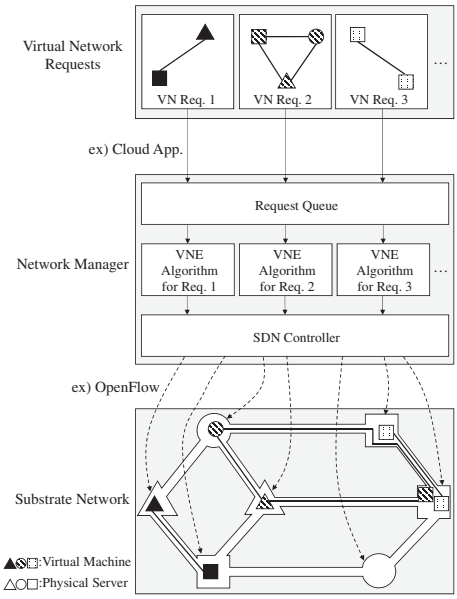

Figure 1: Service model in Software Defined Infrastructure

2 Virtual Network Services in SDI Frameworks

2.1 SDI

Figure 1 describes an service model of SDI frameworks. In the model, customers offer virtual network requests (VN requests) to their service providers. The customers can specify the performance and capacity requirements such as CPU power of a virtual node and bandwidth of a virtual link. They may also specify memory capacity (RAM), storage capacity (HDD), and in some cases, specify the detail of restrictions: OS (Operating System) of the virtual machine, RAID type of a storage, and RAM type of a switching device. We call these specification of virtual nodes as "node attributes".

The service provider has a network manager to handle VN requests. The network manager plays three roles. First, the network manager receives VN requests from customers and push them into a queue. Second, the network manager executes a certain VNE algorithm for each VN request in the queue by a FIFO (First In First Out) principle. The VNE algorithm decides a VN mapping; virtual node mapping and virtual link mapping. Virtual node mapping decides the location of the physical node for each virtual node. Then, the virtual node is hosted on the physical node as a form of virtual machine. Virtual link mapping decides the path on physical network for virtual links between virtual nodes. Lastly, the network manager offers the mapping request to the SDN controller. Then, the service provider installs virtual machines into physical servers and allocates requested computing resources.

2.2 Virtual Network Embedding Problem

Virtual Network Embedding (VNE) is one of the important problems in allocating physical resources to the requested VN request. The physical resources, including resources of the physical network and resources of physical servers, forms a substrate network. OpenStack, which is one of the most general IaaS (Infrastructure as a Service) frameworks, defines virtualized resource components [4]. The substrate node is classified into three types; computing servers, network switches, and storages. Each virtual node may have an individual feature such as supported OS, protocols and storage types.

It is necessary to strictly check the consistency of the node features when embedding a virtual node to a substrate node. That is, the requested features of the virtual node must be supported by the substrate node. To simplify the service model, this paper abstracts the classifications of features of OpenStack into "node attributes".

The mapping of the virtual network has an effect on many aspects such as resource utilization, blocking rate, revenue, QoE, energy efficiency and migration cost. That is why VNE problem deserves considering.

2.3 Centralized Approaches for VNE

A number of approaches coping with VNE problem have been proposed. Most of them try to formulate and solve optimizing problems and maximize/minimize some performance objectives. However, existing VNE formulations are Integer Linear Programing (ILP) and VNE problem is known as $\mathcal{NP}$-hard problem. Thus, some heuristic methods are also developed. Note that both the ILP methods and heuristic methods assume to collect information of the network in advance.

Chowdhury et al. deal with VNE problem to embed multiple VN requests onto a substrate network [3]. They give a formulation as Mixed Integer Linear Programing (MILP) to minimize embedding cost while achieving the balance of resource utilization. Guerzoni et al. formulate as a MILP considering node attribute to maximize the revenue while minimizing resource utilization [4]. Chen et al. present a virtual node mapping method to optimize energy efficiency, and also propose its heuristic algorithm [5]. Fajjari et al. minimize the running cost of the network infrastructure by releasing unused bandwidth of a VN for other VNs [6].

3 Yuragi-based Virtual Network Embedding Method

3.1 Yuragi Principle

Yuragi is a principle that biological organisms adapt to environmental fluctuations. Attractor selection is a model which represents Yuragi principle. The model describes dynamics of state variables x_i $(i = 1, 2, \ldots, n)$ through environmental fluctuations as,

$$\frac{d\mathbf{x}}{dt} = \alpha \times f(\mathbf{x}) + \eta, \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$ represents the system state, activity α is comfortableness of the present system state, $f(\mathbf{x})$ defines deterministic behavior governed by the attractor structure and η represents the stochastic behavior. When the system is in a comfortable state and hence activity α is high, the deterministic term $f(\mathbf{x})$ controls the dynamics while noise η is almost negligible. When the system condition gets worse and α gets close to zero, $f(\mathbf{x})$ is no longer influential and the stochastic term η relatively becomes dominant. Therefore, the system changes its state at random and searches for another attractor. Once the system reaches a new attractor, the activity recovers to a high value and the system will stay in the new good state.

A system driven by Yuragi principle achieves adaptability to environmental changes. The adaptability has two aspects. First, the system is robust to a small fluctuation in the surrounding environment. As long as activity remains higher than a certain level, the system keeps staying in an equilibrium point even though there always exists the noise term. Second, the system has a flexibility to drastic changes in the environment. When the system falls into an uncomfortable state, the activity decreases immediately and the dynamics of the system behavior gets free from the attractor structure.

3.2 Yuragi-based VNE Method

Yuragi-based VNE consists of two phase; attribute-aware virtual node mapping and shortest path virtual link mapping. To begin with, the relation between state variables in Yuragi principle and VNE problem is explained.

Our method decides where to allocate virtual node of attribute k. In other words, the method finds a coupling of attribute k and physical node n. Let the number of attributes be K and the number of physical nodes be N, we prepare variables $\mathbf{x} = (x_1, \ldots, x_{kn}, \ldots, x_{KN})$. A variable x_{kn} is a decision variable whether physical node n becomes a candidate for virtual node with attribute k. Then, the dynamics of each x_i $(i = 1, 2, \ldots, KN)$ is described as,

$$\frac{dx_i}{dt} = \alpha\left\{\varsigma\left(\sum_j W_{ij}x_j\right) - x_i\right\} + \eta, \tag{2}$$

where $\varsigma\left(\sum_j W_{ij}x_j\right) - x_i$ represents a deterministic term and η is a stochastic term. In the first term, the matrix $\mathbf{W}$ represents an attractor structure and is being mentioned later. The function $\varsigma(z)$ is a sigmoid function defined as,

$$\varsigma(z) = \tanh(\frac{\mu}{2}z), \tag{3}$$

where μ represents the gradient in the vicinity of the threshold. Here, the threshold is 0 and the output value of $\varsigma(z)$ gets close to 1 or -1. Note that the range of x_i is $[-1, 1]$. The second term η in Eq. (2) is a random value according with a normal distribution. If $x_i > 0$ and i's corresponding node (attribute) is n (k), physical node n becomes a candidate for the virtual node with attribute k. If x_i $(= x_{kn}) < 0$, the virtual node with attribute k is not embedded to physical node n. Each of the virtual nodes with attribute k is allocated onto one of the candidate nodes in descending order of x_{k*} values. Note that, when physical node n is not compatible with attribute k due to the attribute restriction, x_{kn} is set to 0 without calculating the differential equation (2).

Finally, our method assigns the shortest path for each virtual link request. In this paper, we consider the shortest path routing to minimize hop length on the physical topology. Other routing policies can be applied but is not examined in the evaluation in Sec. 4.

3.2.1 Activity Function with Performance Profile

Activity α is a feedback from the system which reflects the comfortableness of the VN. Let p as an objective metric which is expected to be small, activity is described as,

$$\alpha = \frac{\gamma}{1 + \exp(\delta(p - \theta))}, \tag{4}$$

where γ represents the scale of the activity value, δ represents the gradient around the threshold θ. Note that the activity is subject to be reduced to 0 regardless of Eq. (4). Let V_k be the number of virtual node requests with attribute k, the activity α is reset to be 0 when the number of candidates $|x_{k*}|$ $(s.t.\ x_{k*} > 0)$ is less than V_k. This is necessary because the system state found by Yuragi does not have a sufficient number of candidate nodes. Also, when the available capacity of the physical resource is not enough to embed the

found system state, α is forced to be 0.

In our method, the objective metric p can be directly monitored. However, when the monitoring takes some overhead or it is difficult to monitor p directly in some reasons, the activity should be calculated by estimated p rather than actual p. For the estimation, we consider making use of a performance profile. The profile database consists of the correspondences of delay and resource utilization based on experienced history and is maintained as a table or in some form. In this paper, we consider the end-to-end delay by applying Yuragi principle.

3.2.2 Attractor Structure

The matrix $\mathbf{W}$ in Eq. (2) represents attractor structure. It stores some equilibrium points of virtual node mapping, and the equilibrium point is called attractor. Each attractor is defined as $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_{KN})$ where $x_i \in \{-1, 0, 1\}$. If physical node n is one of the candidates for a virtual node with attribute k, x_{kn} is set to 1. If node n cannot allocate attribute k due to the node attribute restriction, x_{kn} is set to 0. Otherwise, x_{kn} is set to -1. Let M be the number of attractors stored in $\mathbf{W}$, a set of attractors $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$ can be stored by,

$$\mathbf{W} = \mathbf{X}^+\mathbf{X}, \tag{5}$$

where $\mathbf{X}^+$ is the pseudo inverse matrix of $\mathbf{X}$. This way of storing attractors uses the knowledge of Hopfield neural network of associative memory. When the present state is in one of the attractors, $d\mathbf{x}/dt$ in Eq. (2) becomes close to 0 and stay in the attractor.

4 Evaluation with Computer Simulation

This section presents evaluation results of Yuragi-based VNE method by computer simulations.

4.1 Simulation Environment

The substrate network consists of physical servers and links. The number of physical servers (physical nodes) is 50. Each node has capability of hosting virtual node with one of the node attributes. In this environment, each node has three kinds of resource capacities for required virtual machines; CPU, memory and storage capacities of each node are determined uniformly within $[50, 100]$. For each pair of physical nodes, a physical link is prepared with a probability of 0.5. As a result, we obtain the physical topology with 50 nodes and 617 links. The capacity of physical link is determined uniformly within $[50, 100]$. During the simulation, the substrate network is fixed.

Several requests of virtual network are generated and arrived. During the simulation, the number of VN requests is set to 20 and the number of node attributes K is set to 4. Each VN request is generated as follows. The number of virtual machines (virtual nodes) is determined uniformly within $[2, 5]$. Each virtual node belongs to an attribute, and each virtual node requires capacities for CPU, memory, and storage. Each of the required capacities is determined uniformly within $[1, 10]$. Virtual links are non-directed, and each pair of virtual nodes is connected with the probability of 0.5. The number of virtual links is within $[1, 10]$ because the number of virtual nodes is 2 to 5. Each virtual link has required bandwidth, and the required capacity is determined uniformly within $[1, 25]$.

Every 100-time steps, all the 20 VN requests are regenerated in the same way as described above. In addition, at every 10 time step, each VN request fluctuates with relatively small changes: we change the requested capacities by a random integer following the normal distribution with $\mu = 0$ and $\sigma^2 = 1$.

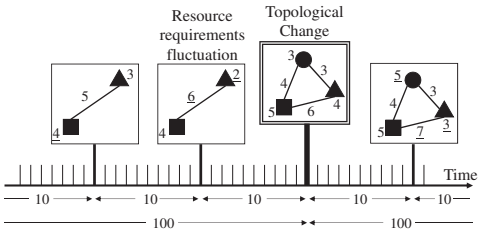

Figure 2: Environmental fluctuations with a lapse of time

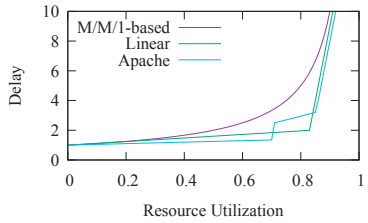

Figure 3: Delay models used for the computer simulation

4.2 Delay Profile

We use end-to-end delay as the objective metric. In an actual environment, the experienced delay may be available by monitoring of packet arrivals. However, when the monitoring takes some overhead or it is difficult to monitor directly for some reasons, the activity should be calculated by estimated p rather than actual p. For the estimation, we consider making use of a performance profile. In the simulation environment, however, estimated delays are used. The estimated value is calculated with some delay models. This paper constructs three delay models as a function of resource utilization. Figure 3 shows the delay models. The first model is the M/M/1-based model which is a basic model of delay in networks. It is derived from a queueing theory model. The second model, which we call the Linear model, is quite simple; the delay increases linearly in accordance with resource utilization. However, the delay increases rapidly at the high load. The third model, which is the Apache model, imitates response time of a web server. As shown in Fig. 3, the model is characterized by the two-stage elevations of delay characterize the model. In the case of web service, the elevations are caused by swapping memory pages and storage disks in accordance with virtual memory architecture.

In the following simulation, d_{ij}, which represents the delay from virtual node i to virtual node j, is calculated as,

$$d_{ij} = w_c \cdot d_i^c + w_m \cdot d_i^m + w_s \cdot d_i^s + w_b \cdot d_{ij}^b, \tag{6}$$

where d_i^c, d_i^m and d_i^s are the computing delay in virtual machine i spent by CPU, memory and storage respectively. Then d_{ij}^b is the propagation delay through virtual link from i to j, and w_* are the weight parameters. Each d_i^* follows the delay model and calculated by its own utilization of physical resources. Note that d_{ij}^b is calculated by utilization of the bottleneck link. The bottleneck link is the physical link that indicates the highest utilization among the path of the virtual link.

Network managers maintain delay profiles in which correspon-

dences between resource utilization and actual measured delay are recorded. Referring the delay profile make it possible for them to estimate delays in VN supposing the VN request to be embedded. In this simulation, the profile is assumed to follow the delay functions in Fig. 3.

4.3 Heuristic Method for Comparison

As for the benchmark of our method, a heuristic VNE method is also simulated in the same environments. The heuristic method is composed of two phases; virtual node mapping based on greedy algorithm [8] and virtual link mapping on shortest paths. The VNE method executes the following algorithm for one VN request after another.

(1) Execute the following processes for each virtual node v.

(1.1) Find the set S of physical nodes that accept the attribute of v and have enough vacant of resource capacities to embed v. When S is null, reject the VN request and finish.

(1.2) Find the physical node that indicates the highest value of H among S, where H is defined as Eq. (7). Then reserve the resources of that physical node.

(2) Embed the virtual nodes according to the reservation taken in (1).

(3) For each virtual link between virtual nodes embedded in (2), find a path that is the minimum hop in physical topology. Embed the virtual links onto the paths. When the shortage of link bandwidth occurs, reject the VN request.

The greedy method aims to minimize utilization of node and link resources. The heuristic method calculates an available resource indicator H for each physical node n defined as,

$$H(n) = C_n \times M_n \times S_n \times \sum_{l \in L(n)} B_l, \qquad (7)$$

and avoids embedding a virtual node onto bottleneck resources. C_n, M_n and S_n represent the available capacity of CPU, memory and storage on physical node n. The set $L(n)$ represents a set of physical links attached to node n, and B_l represents the available capacity of physical link l. The computational complexity is $\mathcal{O}(n \log n)$ for sorting $H(n)$, assuming the shortest path between every node pairs is available in advance.

4.4 Simulation Results

We first show the adaptability of Yuragi-based VNE method with M/M/1-based delay model. The threshold of activity θ in Eq. (4) is set to 2.0 with regarding the metric p as the maximum of d_{ij} for every pair of virtual nodes i and j. The weight values in Eq. (6) are set as $w_c = w_m = w_s = w_b = 0.25$. In the simulation, the Yuragi-based method calculates the VN mapping at each time step and migrates the VN until the system state converges to an attractor. The greedy method executes VN migration according to the demand changes at every 10 time steps.

4.4.1 Adaptability of Yuragi-based Method to Request Fluctuation

Figure 4 shows the maximum delay on a VN request out of the 20 requests. The activity of each VN is also shown in Fig. 5. In the figure, the region denoted as "Failure" represents a failure of embedding the VN caused by the shortage of physical resources or violation of other restrictions. Note that the demands of VN requests fluctuate with relatively small changes at every 10 time steps and the

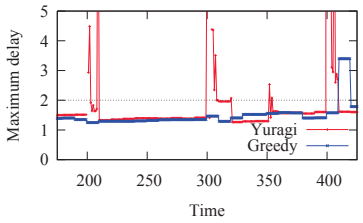

Figure 4: Maximum delay on a VN

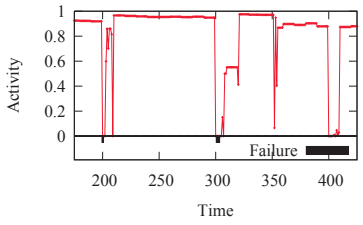

Figure 5: Activity on a VN

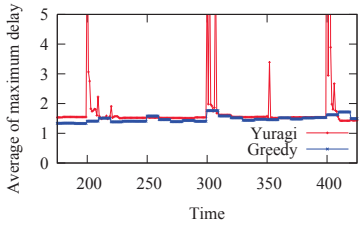

Figure 6: Average of maximum delay for 20 VNs

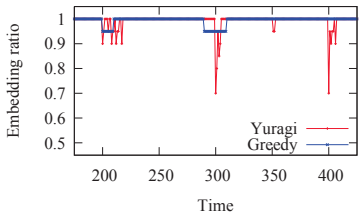

Figure 7: Embedding ratio of VN requests

demands fluctuate greatly at every 100 time steps. Thus, the maximum delays for Yuragi-based method exceed the threshold drastically at every 100 time steps owing to the topological changes in VN requests. The activities drop sharply, and then the VN migration starts. Within few steps, the activities get recovered and converge to another system state. Against a small fluctuation of required capacities occurred at every 10 time steps, VN migrations occurs only if the activities decrease extremely, for instance, as shown in time step 350 in Fig. 4. Figure 6 shows the mean of maximum delay of 20 VN requests. The Yuragi-based method does not achieve as small delay as the greedy method in general. This is because the Yuragi-based method does not aim to minimize the delay or the resource utilization but keep them smaller than a certain threshold. Making the threshold smaller might achieve smaller delay, but will result in taking longer convergence time to find an attractor.

Figure 7 shows the embedding ratio indicating how many VN requests are accepted out of the 20 requests. The topological changes occurred at every 100 time steps cause a temporal decrease in the embedding ratio, but both of the methods keep almost 95% to 100% acceptances excluding that.

Figure 8 shows the number of VN migrations defined as the num-

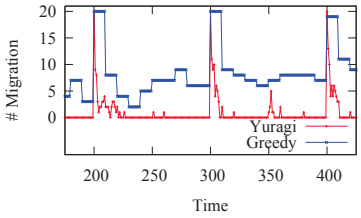

Figure 8: The number of VN migrations

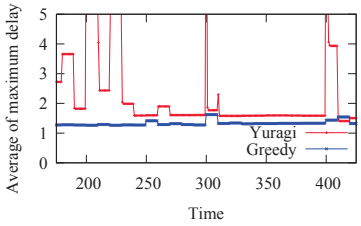

Figure 9: Average of maximum delays for 20 VNs: Linear delay model

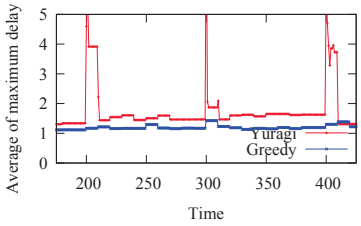

Figure 10: Average of maximum delays for 20 VNs: Apache delay model

ber of the VNs whose location has been changed from the previous operation. Note that the operations are performed at every step by the Yuragi-based method, at every 10 steps by the greedy method. When the VN requests are regenerated at every 100 steps, almost all VNs are migrated for the both methods. The Yuragi-based method migrates a few VNs for a while but requires a few number of VN migrations to the small fluctuations. The greedy method migrates 5 to 10 VNs every time for the required capacity fluctuations. This is because that the greedy method tries to achieve the better objective values even though the improvement in delay is marginal. Against the small fluctuation at every 10 time steps, the number of VN migrations by the Yuragi-based method is totally 95 for 500 time steps simulation, and 171 by the greedy method. This result indicates that the Yuragi-based method adapts to demand fluctuations with about 45% fewer VN migrations than the greedy method.

4.4.2 Availability of Various Models for Delay Profile

The Yuragi-based method is available for other delay profiles than M/M/1-based model. The M/M/1-based model is constructed by a basic queueing theory in a communication network. However, it is likely that the delay profiles in SDI frameworks will be beyond the conventional queueing theory owing to its unique network structure with virtual machines and computational bottleneck on them. Thus, other delay profiles than M/M/1-based model are also considered here. Figure 9 shows the mean of maximum delays for the Linear model, and Fig.10 for the Apache model. For the both models, the Yuragi-based method shows trends similar to the behavior for M/M/1-based model. Thus, the Yuragi-based method can be applied without grasping bottom causes of the delay as long as the record of experienced delays is available.

5 Conclusion

This paper presented a VNE method based on Yuragi principle in SDI frameworks. A system driven by Yuragi principle achieves adaptability to environmental changes, and the dynamics is described as the attractor selection model. In attractor selection model, the system behavior is governed by an activity and a small perturbation. When activity is high, the control state of the system is in a good condition and stay in that state. When activity gets low or the condition gets uncomfortable by environmental changes, the system looks for another stable state. The Yuragi-based VNE method decides the mapping of virtual nodes by means of attractor selection, where the network mapping is regarded as the system state and the activity is defined as a certain performance objective. In the evaluation, we consider the end-to-end delay as the activity. Simulation results show that the method satisfies smaller delay and adapt to the request fluctuations by rearranging the VN mapping for drastic changes in environments. The Yuragi-based method decreases VN migrations by about 45% than a heuristic method to adapt the fluctuations in required resource capacities.

In the future work, evaluation for other situations or other activity definition should be done to confirm the effectiveness of the method. We will also investigate a method of constructing the attractor structure to improve the convergence time or some performances.

Acknowledgements

This research was supported in part by a Grant-in-Aid for Scientific Research (A) (No. 15H01682) from the Japan Society for the Promotion of Science (JSPS) in Japan.

References

[1] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future Internet," *Computer Networks*, vol. 75, Part A, pp. 453–471, Dec. 2014.

[2] S. Sezer, S. Scott-Hayward, P. 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, pp. 36–43, July 2013.

[3] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proceedings of IEEE INFOCOM*, pp. 783–791, Apr. 2009.

[4] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A novel approach to virtual networks embedding for SDN management and orchestration," in *Proceedings of IEEE NOMS*, pp. 1–7, May 2014.

[5] X. Chen, C. Li, and Y. Jiang, "Optimization model and algorithm for energy efficient virtual node embedding," *IEEE Communications Letters*, vol. 19, pp. 1327–1330, Aug. 2015.

[6] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Adaptive-VNE: A flexible resource allocation for virtual network embedding algorithm," in *Proceedings of IEEE GLOBECOM*, pp. 2640–2646, Dec. 2012.

[7] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Adaptive virtual network topology control based on attractor selection," *IEEE Journal of Lightwave Technology*, vol. 28, pp. 1720–1731, June 2010.

[8] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 17–29, Mar. 2008.