Master's Thesis

Title

Yuragi-based Virtual Network Embedding Method for Software Defined Infrastructure with Uncertain Environments

Supervisor

Professor Masayuki Murata

Author

Koki Inoue

February 12th, 2016

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Yuragi-based Virtual Network Embedding Method for Software Defined Infrastructure with Uncertain Environments

Koki Inoue

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.

Keywords

Virtual Network Embedding
Software Defined Infrastructure

Yuragi-based VNT control

Software Defined Networks

Network Function Virtualization

Virtual Node Mapping

Delay Profile

Contents

List of Figures

1 Introduction

Information network faces with new emerging services such as mobile services, cloud computing services, and social services. To deploy new services on information network and/or information systems immediately, SDI (Software Defined Infrastructure) has been focused. SDI provides virtualized infrastructures to customers by slicing computing resources and network resources. In the SDI framework, thanks to the advance of virtualization technologies combined with software technology, the customers order resource requests to service providers via, e.g., web applications, and the sliced virtualized resource is immediately assigned to the customers.

A key to enjoying the SDI framework is the network virtualization technologies and their control. The network virtualization technologies are at the stage of research and development phase. 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. SDN/NFV technologies enable programmable and automated network control while the conventional systems require the network operator to configure various kinds of network devices [1–5]. That is, SDI frameworks realized by SDN/NFV technologies have a potential to support a rapid and flexible deployment of services such as on-demand resource allocation, self-service provisioning, and secure cloud services [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 [6–12]. The VNE problem is a placement problem to allocate virtual resources to the physical network with the optimization of some performance objectives. In the VNE problem, service demands from customers are translated to VN (Virtual Network) requests. VN consists from the virtual nodes and the virtual links. Each of the virtual nodes is hosted on a physical node as a form of virtual machines. Then, the virtual nodes are connected through a path of the physical node, which forms virtual links.

The VNE problem is divided into two sub-problems; virtual node mapping and virtual link mapping. Virtual node mapping decides the location of the physical node for each virtual node. Note that each virtual node must be allocated to a physical node supporting its "node attribute". Node attribute is a classification of nodes defined by supported OS, storage types, or a use of the node for, e.g., computing, storage, or packet switching. Virtual link mapping decides the path on physical network for virtual links between virtual nodes.

In [9–12], 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. As the network operators want to know the current situation more correctly and precisely, more information is necessary to collect. This will lead to the occupation of link bandwidth, an increase of delay, and a bottleneck of network scalability [5]. Note that the calculation time to obtain a solution of the optimization problem also gets larger. However, the calculation time may be relaxed by some heuristic algorithm with a little (with optimistic outlook) or more sacrifice of the quality of the solution. 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. For example, Yuragi is used for representing small swaying in the candle. Yuragi is a mechanism of adaptability of organisms and is often 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 [13–16]. 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. One process of the method is executed for each VN slice respectively, and each process needs information only about its VN requests. Each of the processes behaves to improve its own performance function, considering other VNs as a part of an external perturbation, i.e., Yuragi.

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.

2 Virtual Network Services in SDI Frameworks

In this section, we describe SDN frameworks and explain a service model for SDI frameworks. At first, a whole system of the virtual network service is explained. Next, one of the important problems in the SDI service called VNE is described. VNE affects many aspects of the service performance such as resource utilization, revenue, QoE (Quality of Experience) and energy efficiency. Therefore, VNE problem has been a hot topic in network virtualization field. The research trends are introduced in the following.

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 VN request consists of a topology, which is a combination of virtual nodes and virtual links. Then, the provider assigns computing resources on the virtual nodes by preparing virtual machines. Then, the provider configures the packet forwarding rules on the network switches via SDN controller to form virtual links.

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. Then, the virtual nodes are connected through the path. When the VNE

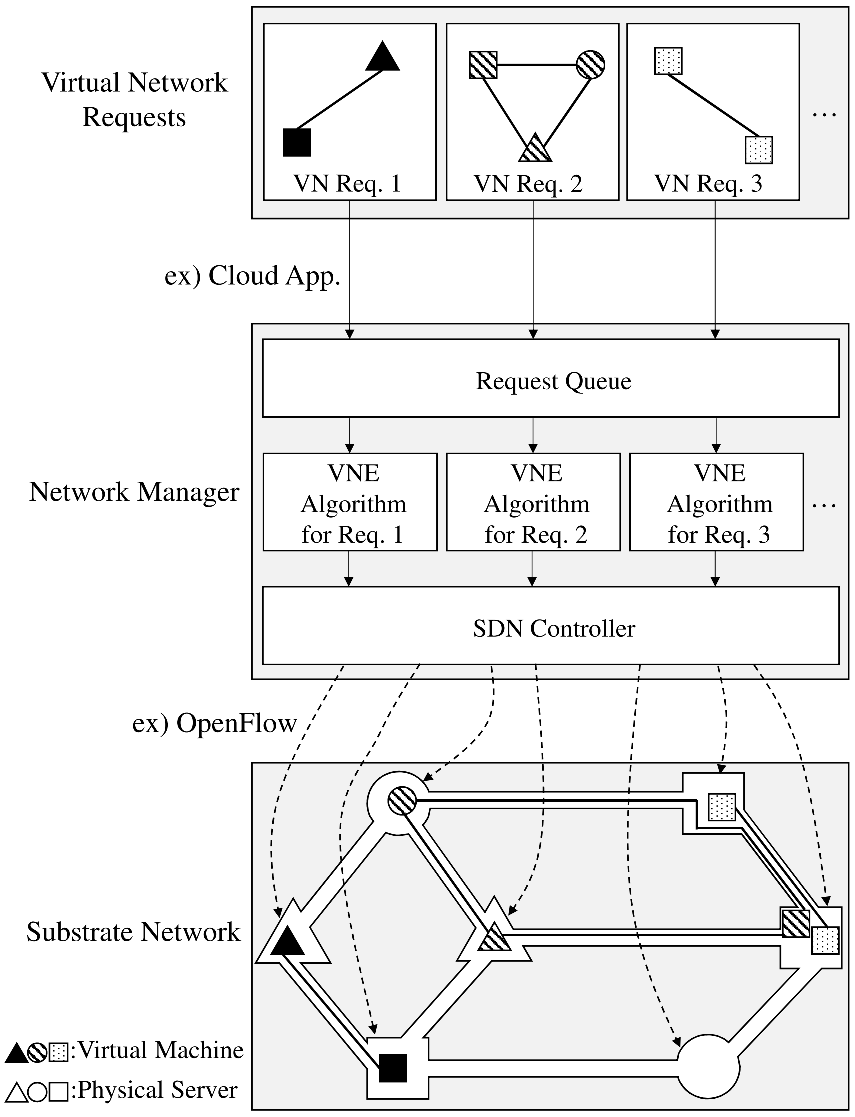


Figure 1: Service model in Software Defined Infrastructure

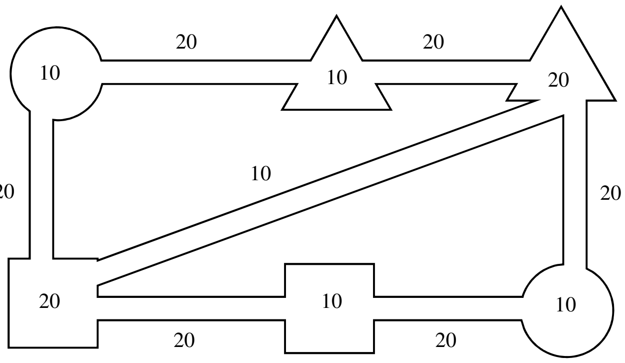
algorithm fails to find a VN mapping due to the shortage of the physical resources, the VN request is rejected. Third and lastly, the network manager offers the mapping request to the SDN controller. Note that the SDN controller might be managed by other organizations such as infrastructure providers rather than the service provider. Then, the service provider installs virtual machines into physical servers and allocates requested computing resources. Then, the SDN controller accesses substrate nodes via a certain protocol such as OpenFlow and reconfigures the forwarding rules to establish the virtual links.

2.2 Virtual Network Embedding Problem

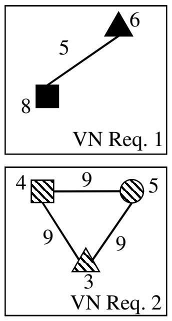
Virtual Network Embedding (VNE) is one of the important problems in allocating physical resources to the requested VN request. The physical resources, including resources of the physical network and resources of physical servers, forms a substrate network. OpenStack, which is one of the most general IaaS (Infrastructure as a Service) frameworks, defines virtualized resource components [10]. 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. Figure 2 describes an illustrative example to show how experienced delay of VNs differs dependent on the mapping of the virtual network. Figure 2(a) shows a substrate network and the numerical values in the figure represent CPU capacities of the physical servers and the link bandwidths. Figure 2(b) shows the VN requests including resource requirements. Figure 2(c) and 2(d) show two patterns of VN mapping, denoted as mapping A and B. In general, the delay on a server gets longer when the CPU utilization is higher, and the delay on a link also gets longer when the link utilization is higher. In the case of mapping A, the CPU utilization on one of the substrate node reaches 80% and the calculation delay gets longer. However, in the case of mapping B, the CPU utilizations are at most 50%. As for the delay on a link, the maximum of link utilization of the substrate link is 90% in the case of mapping A. In the case of mapping B, the link utilizations of the substrate links are low, and the delay will not get longer. Therefore, the experienced delay of mapping B is expected to be shorter than that of mapping A. Thus, mapping B is preferred as for the solution of VNE problem.

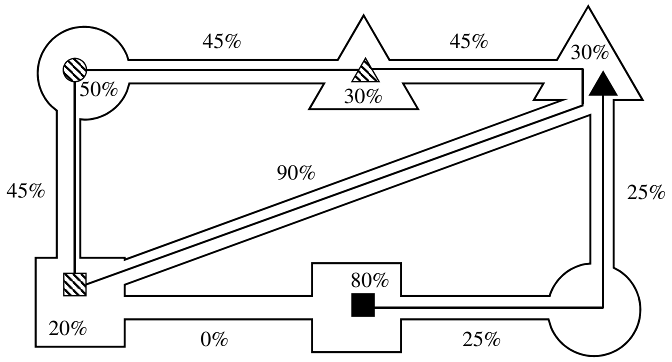
Generally, VNE problem is divided into two phases, i.e., Virtual Node Mapping (VNoM) and Virtual Link Mapping (VLiM) problem. VNoM is to obtain a set of matching be-



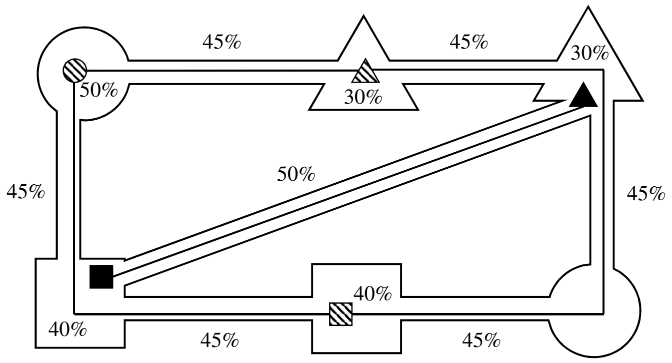
(a) Substrate network



(b) VNs



(c) VN mapping A



(d) VN mapping B

Figure 2: Comprehension of VNE problem with a simple example

tween virtual nodes and substrate nodes under the constraint that the substrate node must support the node attribute of the virtual node. VLiM is to obtain a set of links in the substrate network which connects one virtual node to another virtual node.

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 [9]. 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 [10]. Chen et al. present a virtual node mapping method to optimize energy efficiency, and also propose its heuristic algorithm [11]. Fajjari et al. minimize the running cost of the network infrastructure by releasing unused bandwidth of a VN for other VNs [12].

3 Yuragi-based Virtual Network Embedding Method

This section proposes Yuragi-based VNE method for SDI frameworks. The Yuragi principle, which is often called attractor selection model, explains the biological adaptability. The key concept of attractor selection model is that the system behavior is governed by a single barometer, called "activity", and a small perturbation which we call "Yuragi". The activity is kind of comfortableness for the system, and with a feedback of the activity and the small perturbation, the control state of the system falls into a comfortable state. When activity is high, the control state of the system is in a good condition and stay in that state. Such the equilibrium point is called "attractor". When activity gets low or the condition gets uncomfortable by environmental changes, the system gets out of the attractor and then looks for another attractor with a feedback of the activity and the small perturbation.

The proposed VNE method is expected to enjoy the adaptability of Yuragi to environmental changes. That is, VN migrations are driven according to experienced performances and the new VN mapping is obtained by means of attractor selection. A process of the Yuragi-based method is executed for each VN request. Thus, multiple processes are executed in parallel to deal with multiple VN slices. Different from optimizing problems and their related heuristic, the Yuragi-based method can avoid the necessity of collecting detailed information about the entire network. The process for a VN request only needs an information for comfortableness and does not need any information related to the other VN requests.

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 Performance Objectives

We can select various definitions of the activity when Yuragi principle is applied to the VNE problem. Yuragi-based VNE method tries to find a system state which keeps a high activity. The high activity should be designed so that a performance objective may not violate a required threshold.

The following metrics are considered as performance objectives. *Delay* is the communication delay is caused by propagation delay through links and computing delay on servers or switches. The minimizing delay is a basic requirement for networks, and as a matter of course, the delay is preferred to be small to improve the user experience. *Resource utilization* is link bandwidth utilization and server capacity (CPU, memory, storage) utilization, and is often considered in the VNE problem [17, 18]. When resource utilization is low, there is much space to accommodate future VN requests. On the other hand, less resource utilization leads to an inefficiency of energy consumption. Note that resource utilization and delay are positively correlated. *Revenue* from customers for embedding VNs is a metric that reflects a business. Service providers impose a fee on their customers in proportion to a number of embedded resources such as the number of CPU cores. Service providers also pay a cost to accommodate the VNs by consuming physical resources. Some VNE problems try to maximize the revenue by maximizing the income with the minimum outcome [6, 9, 10, 19]. *Energy consumption* on servers and links also deserves consideration. If any virtual nodes are not embedded in a server, sleeping the server can save energy consumption. The energy consumption of the entire network may be estimated from the number of active nodes and links, and their utilization [11, 20]. Efficient energy consumption leads to reduce running cost and achieves a green ICT [21, 22].

Conventional works usually use link utilization and/or energy consumption as a performance objective because these are described by a linear function of a traffic load. Note that a linearity in a mathematical sense is one of the key factors to solve the optimization problem. In this paper, we focus on user's experienced delay, which is not a linear function, as the performance objective comfortability because the experienced delay is one of the simplest and most fundamental performance objectives in networking. It is true that link utilization is often used as the performance objective of a virtual network control. However, the experienced delay is more important measure in networking and especially important for SDI frameworks. The longer delay causes considerable degradation in QoE of the application which is running on the virtual network. Thus, customers of SDI services want to require a low delay to the infrastructure provider. Moreover, under virtualization environment, the delay is caused by not only utilization of the network bandwidth but also workloads on the virtual machines, and the delay gets extremely large under heavy workloads [23, 24]. Therefore, it is difficult to deal with delay requirements in conventional approaches. In this paper, we consider the end-to-end delay by applying Yuragi principle.

3.3 Yuragi-based VNE Method

This section explains our Yuragi-based VNE method. Our proposed method 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.

3.3.1 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,

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

and Fig. 3 depicts the activity function, where γ represents the scale of the activity value, δ represents the gradient around the threshold θ. Let γ be 1, the activity value

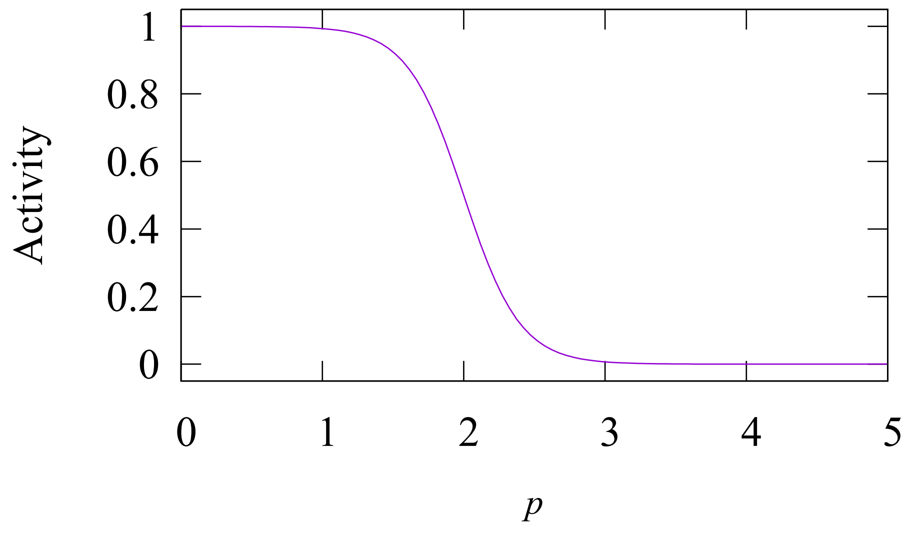


Figure 3: Activity function for $\gamma = 1.0$, $\delta = 5.0$ and $\theta = 2.0$

gets close to 1 if $p < \theta$. Otherwise, the activity becomes 0. 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. For example, the experienced delay is a basic metric of network performance but it can be obtained only when the system is running. However, the delay has a feature correlated to resource utilization. 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. If the present resource utilization and the required capacity are known, an estimated delay can be obtained by referring the profile database instead of running the computing service.

3.3.2 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.

3.4 VN migration

To apply the Yuragi-based VNE method, VN migration is executed as follows to reduce overhead of installing virtual machines and restarting processes:

Step 1 When the activity gets extremely low, the provider suspends virtual machines and take a snapshot of them.

Step 2 The provider executes Yuragi-based VNE method by using the performance profile instead of actual performance. The profile enables to get an estimated performance without running the services. That is, the activity is obtained while the virtual machines are sleeping. Then, the Yuragi method migrates the VN according to the activity.

Step 3 Once the system state converges to a certain VN mapping, the provider restarts the virtual machines and load the snapshots.

Step 4 When the actual performance gets worse, the activity decreases again. Then, the above steps are repeated again.

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 $\sigma^2 = 1$ (see Fig. 4).

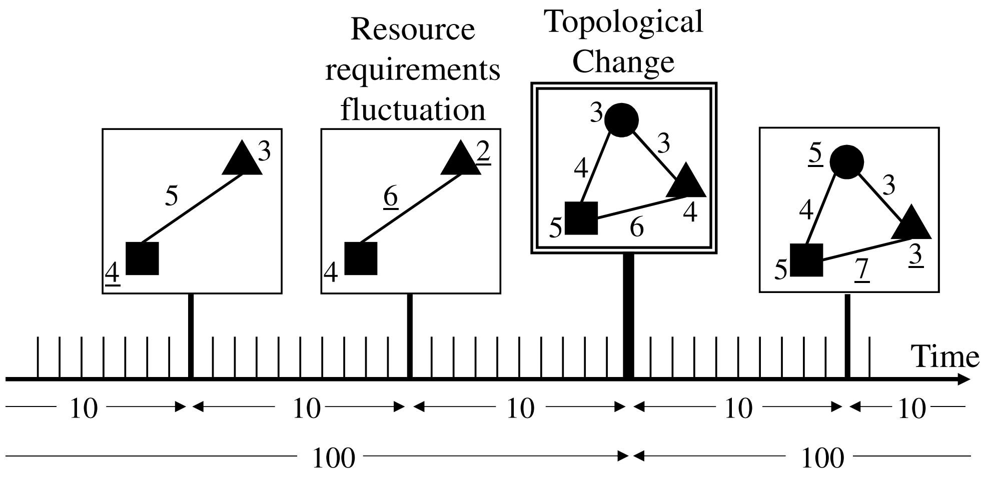


Figure 4: Environmental fluctuations with a lapse of time

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 5 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. 5, 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,

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

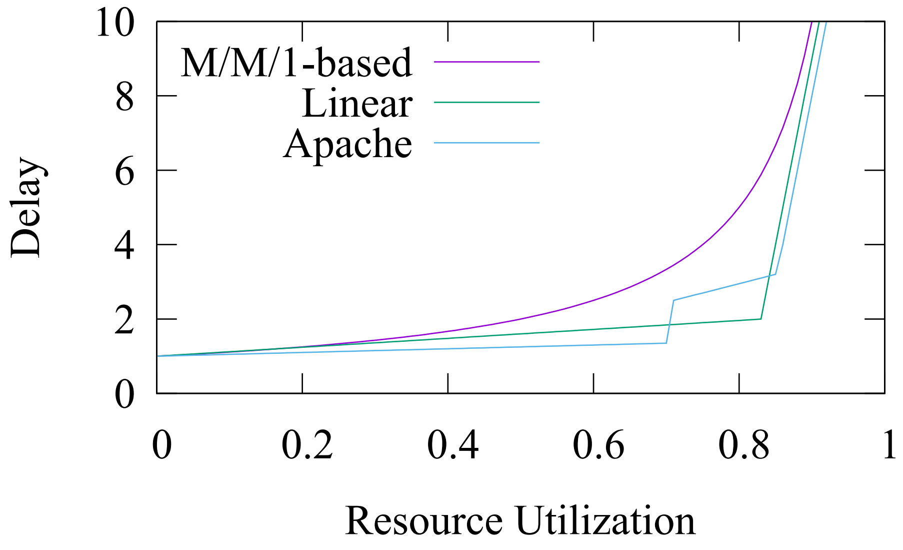


Figure 5: Delay models used for the computer simulation

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 correspondences 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. 5.

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 [25] 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,

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

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

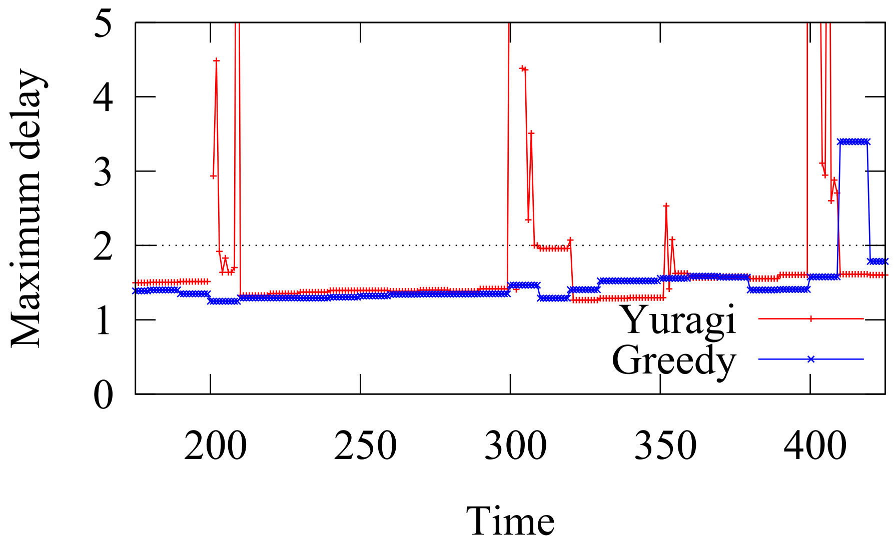
Figure 6 shows the maximum delay on five VN requests out of the 20 requests. The activity of each VN is also shown in Fig. 7. In the figure, the region denoted as “Failure” represents a failure of embedding the VN caused by the shortage of physical resources or violation of other restrictions. Note that the demands of VN requests fluctuate with relatively small changes at every 10 time steps and the 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 the VN with ID 0 (Fig. 6(a) and 7(a)). Figure 8 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 9 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.

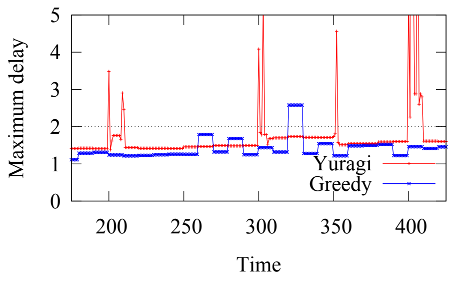
Figure 10 shows the number of VN migrations defined as the number of the VNs whose location has been changed from the previous operation. Note that the operations are performed at every step by the Yuragi-based method, at every 10 steps by the greedy method. When the VN requests are regenerated at every 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

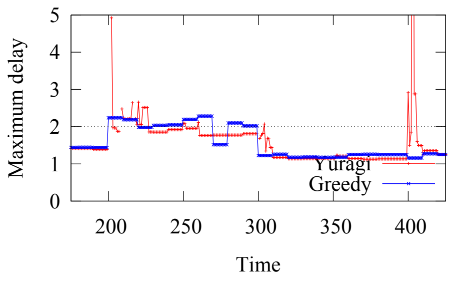
The Yuragi-based method is available for other delay profiles than M/M/1-based model. The M/M/1-based model is constructed by a basic queueing theory in a communication network. However, it is likely that the delay profiles in SDI frameworks will be beyond the conventional queueing theory owing to its 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 11 and 12 show the delay for the Linear model, Fig. 13 and 14 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.



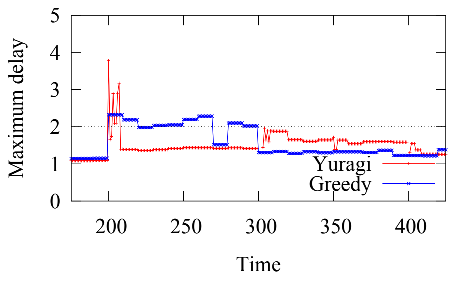
(a) ID 0



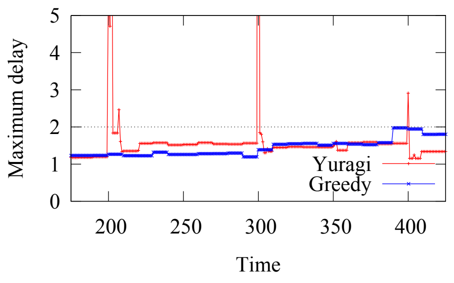
(b) ID 1



(c) ID 2

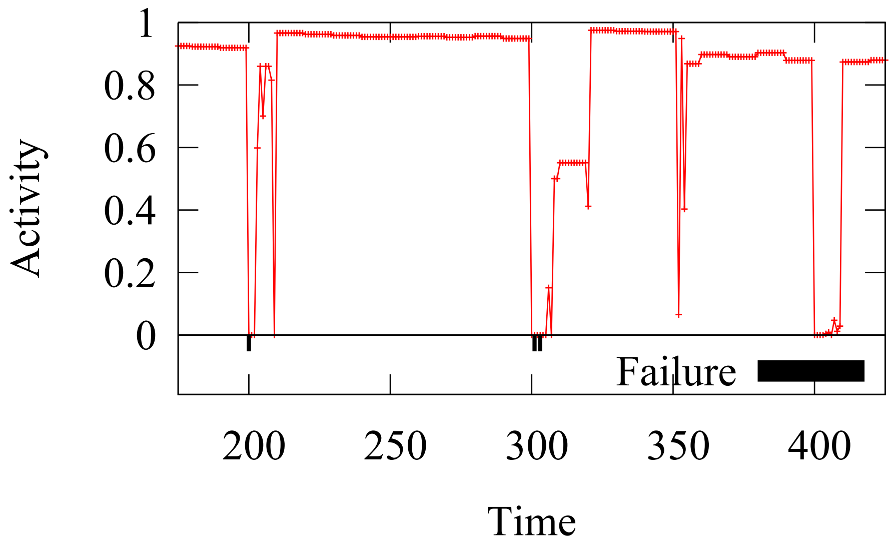


(d) ID 3

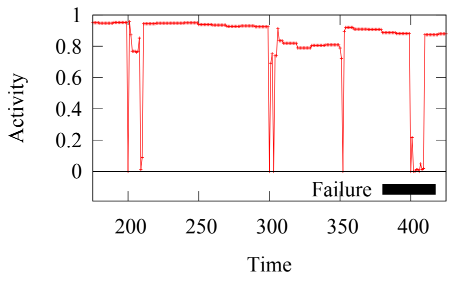


(e) ID 4

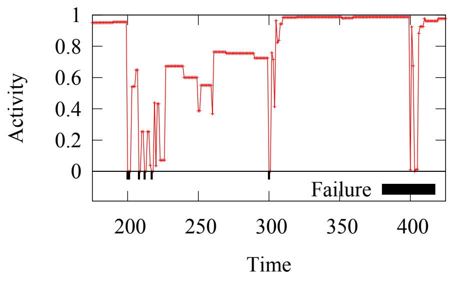
Figure 6: Maximum delay on each VN



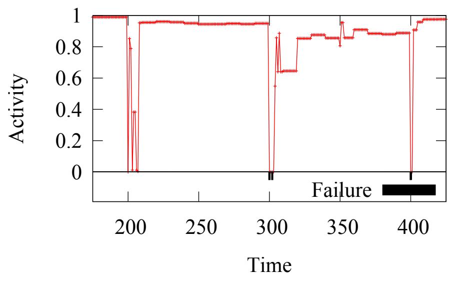
(a) ID 0



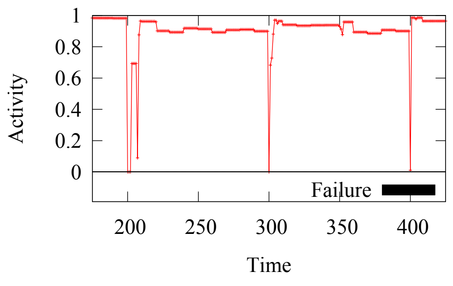
(b) ID 1



(c) ID 2



(d) ID 3



(e) ID 4

Figure 7: Activity on each VN

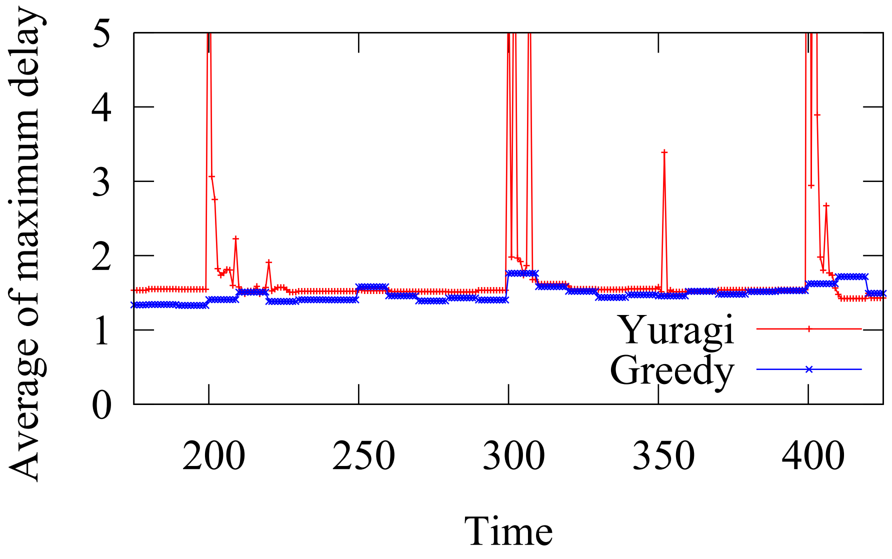


Figure 8: Average of maximum delay for 20 VNs

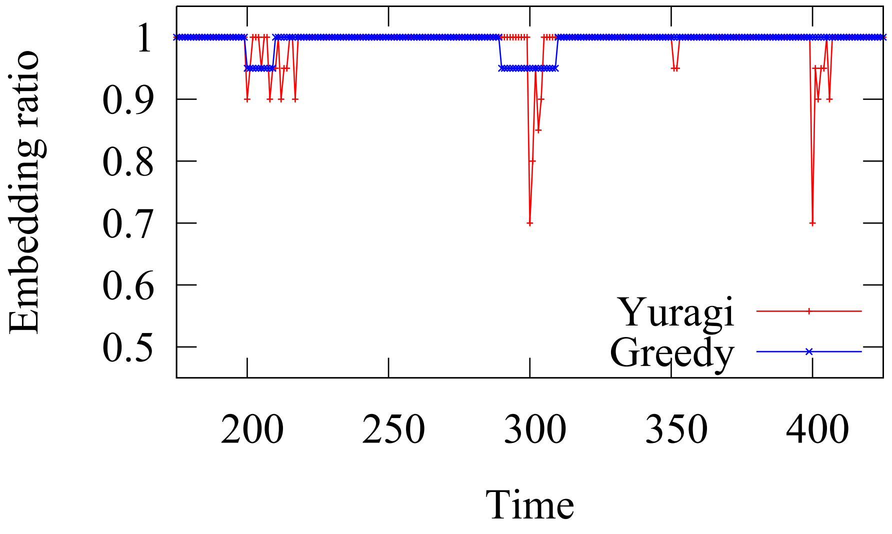


Figure 9: Embedding ratio of VN requests

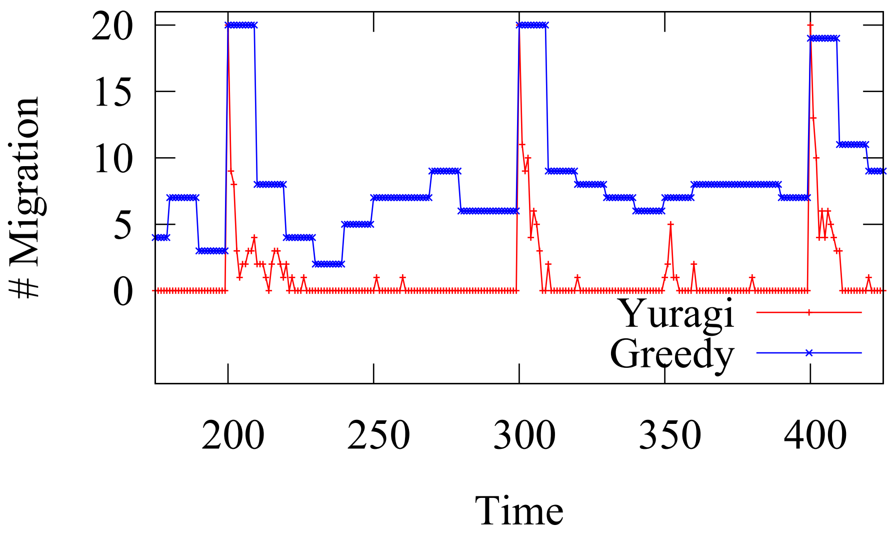
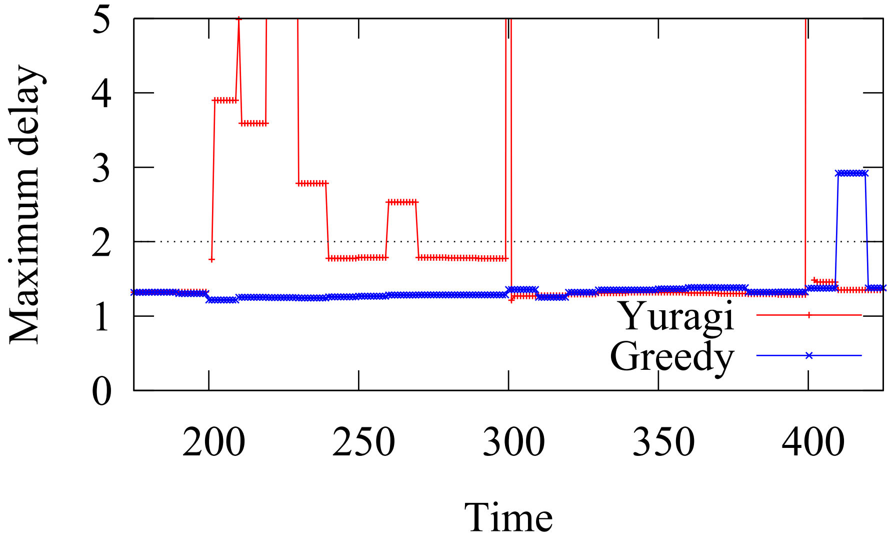
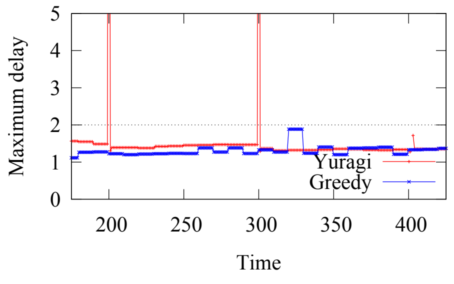


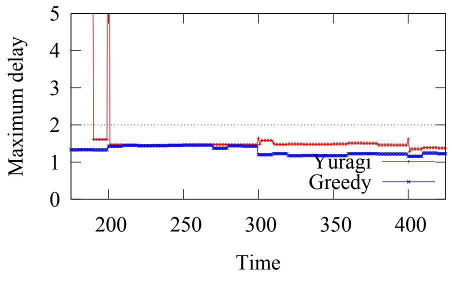
Figure 10: The number of VN migrations



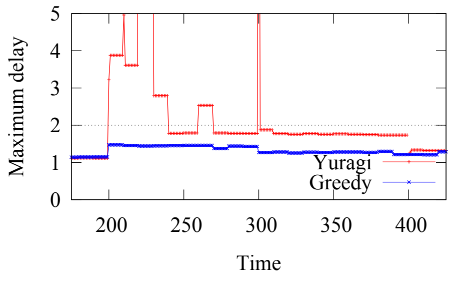
(a) ID 0



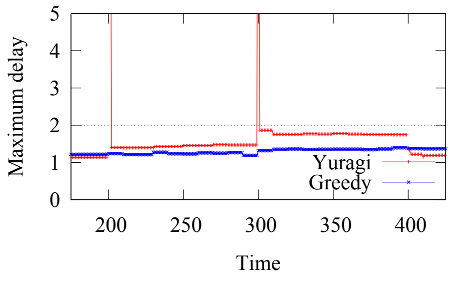
(b) ID 1



(c) ID 2



(d) ID 3



(e) ID 4

Figure 11: Maximum delay on each VN: Linear delay model

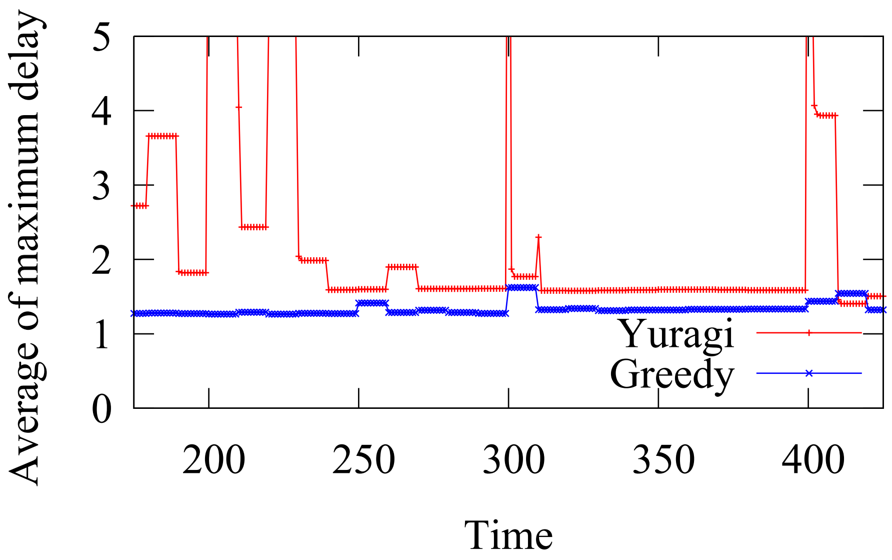
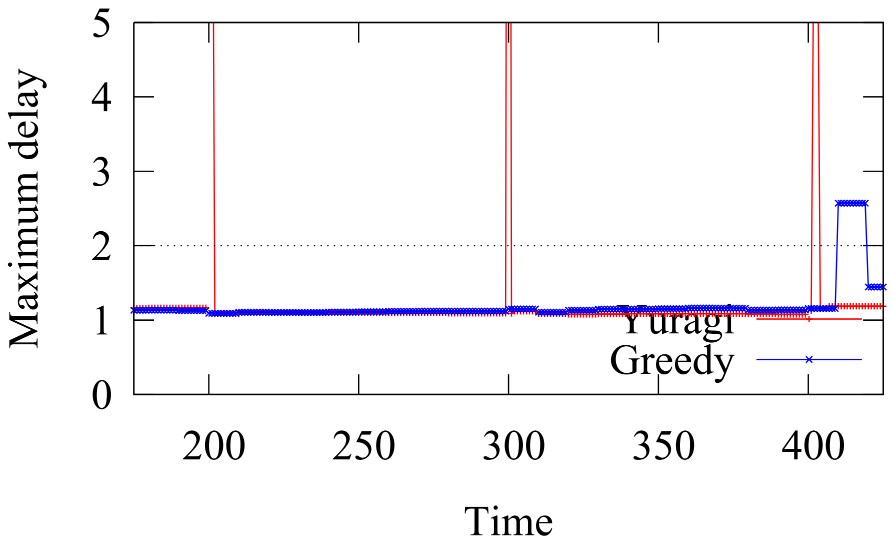
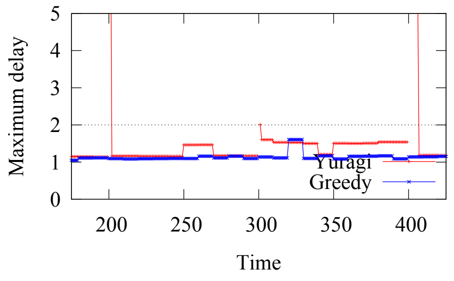


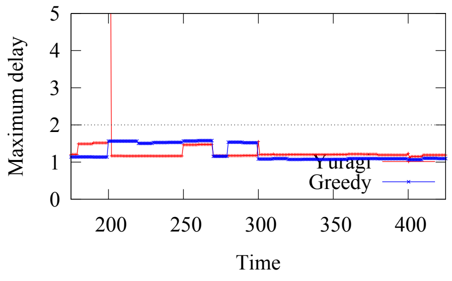
Figure 12: Average of maximum delay for 20 VNs: Linear delay model



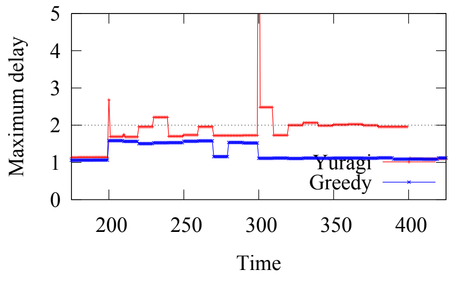
(a) ID 0



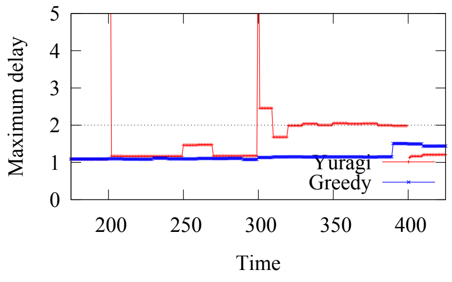
(b) ID 1



(c) ID 2



(d) ID 3



(e) ID 4

Figure 13: Maximum delay on each VN: Apache delay model

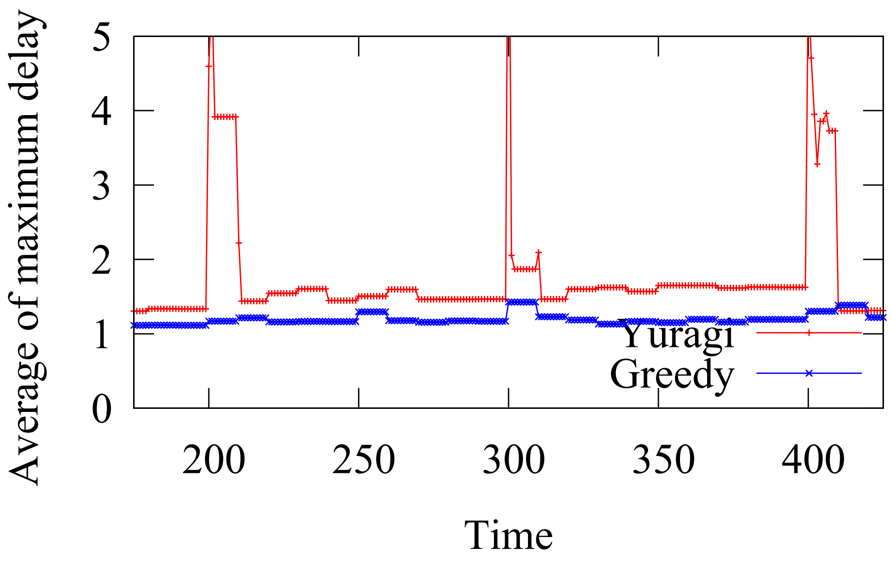


Figure 14: Average of maximum delay for 20 VNs: Apache delay model

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

I really appreciate Professor Masayuki Murata of Osaka University. He has supervised my research activity with his great perspective. I am very honored to have had the amazing opportunity since I was an undergraduate. I am full of gratitude to Associate Professor Shin'ichi Arakawa of Osaka University. We discussed my research many times, and he taught me enormous knowledge and skill to address the problems that I faced. This thesis could never be completed without his support. I would like to express my sincere gratitude to Dr. Satoshi Imai of Fujitsu Laboratories Ltd. for his counsel and support with this research. I am also grateful to him a lot during my research internship at Fujitsu Lab. I am deeply grateful to Assistant Professor Yuichi Ohsita and Assistant Professor Daichi Kominami of Osaka University, for helpful advices. I sincerely thank my senior associates, including Mr. Toshihiko Ohba, for giving me many thoughtful advices and for boosting my enthusiasm to the research activity. Finally, I would like to thank all the members of Advanced Network Architecture Research Laboratory of Osaka University for their help and encouragement, and I will not forget the days we worked together.

References

[1] C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, “Segment routing architecture,” *Internet draft* `draft-ietf-spring-segment-routing-06.txt`, Oct. 2015. work in progress.

[2] 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.

[3] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1617–1634, Feb. 2014.

[4] P. Bhaumik, S. Zhang, P. Chowdhury, S. S. Lee, J. Lee, and B. Mukherjee, “Software-defined optical networks (SDONs): A survey,” *Photonic Network Communications*, vol. 28, pp. 4–18, June 2014.

[5] 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.

[6] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, “Virtual network embedding through topology-aware node ranking,” *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 38–47, Apr. 2011.

[7] J. Lischka and H. Karl, “A virtual network mapping algorithm based on subgraph isomorphism detection,” in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, pp. 81–88, Aug. 2009.

[8] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, “Virtual network embedding: A survey,” *IEEE Communications Surveys Tutorials*, vol. 15, pp. 1888–1906, Feb. 2013.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] K. Mizumoto, S. Arakawa, Y. Koizumi, D. Shimazaki, T. Miyamura, S. Kamamura, K. Shiomoto, A. Hiramatsu, and M. Murata, "A distributed control of virtual network topologies by using attractor selection model," in *Proceedings of NOLTA*, Oct. 2012.

[15] S. Kamamura, Y. Koizumi, D. Shimazaki, T. Miyamura, S. Arakawa, K. Shiomoto, A. Hiramatsu, and M. Murata, "Attractor selection-based virtual network topology control with dynamic threshold reconfiguration for managed self-organization network," in *Proceedings of the 24th International Teletraffic Congress*, pp. 1–6, Sept. 2012.

[16] T. Ohba, S. Arakawa, Y. Koizumi, and M. Murata, "Scalable design method of attractors in noise-induced virtual network topology control," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, pp. 851–863, Sept. 2015.

[17] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, “VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic,” in *Proceedings of IEEE ICC*, pp. 1–6, June 2011.

[18] S. Zhang, Z. Qian, J. Wu, and S. Lu, “An opportunistic resource sharing and topology-aware mapping framework for virtual networks,” in *Proceedings of IEEE INFOCOM*, pp. 2408–2416, Mar. 2012.

[19] L. Gong, Y. Wen, Z. Zhu, and T. Lee, “Revenue-driven virtual network embedding based on global resource information,” in *Proceedings of IEEE GLOBECOM*, pp. 2294–2299, Dec. 2013.

[20] J. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, “Energy efficient virtual network embedding,” *IEEE Communications Letters*, vol. 16, pp. 756–759, May 2012.

[21] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, “Using ant colony system to consolidate VMs for green cloud computing,” *IEEE Transactions on Services Computing*, vol. 8, pp. 187–198, Mar. 2015.

[22] D. Bruneo, A. Lhoas, F. Longo, and A. Puliafito, “Modeling and evaluation of energy policies in green clouds,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 3052–3065, Nov. 2015.

[23] J. Whiteaker, F. Schneider, and R. Teixeira, “Explaining packet delays under virtualization,” *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 38–44, Jan. 2011.

[24] G. Wang and T. Ng, “The impact of virtualization on network performance of Amazon EC2 data center,” in *Proceedings of IEEE INFOCOM*, pp. 1–9, Mar. 2010.

[25] 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.