PAPER *Special Section on Network Systems for Virtualized Environment*

Virtual Network Allocation for Fault Tolerance Balanced with Physical Resources Consumption in a Multi-Tenant Data Center

Yukio OGAWA[†a)], **Go HASEGAWA**[††], ***Members***, ***and*** **Masayuki MURATA**[††], ***Fellow***

SUMMARY In a multi-tenant data center, nodes and links of tenants' virtual networks (VNs) share a single component of the physical substrate network (SN). The failure of a single SN component can thereby cause the simultaneous failures of multiple nodes and links in a single VN; this complex of failures must significantly disrupt the services offered on the VN. In the present paper, we clarify how the fault tolerance of each VN is affected by a single SN failure, especially from the perspective of VN allocation in the SN. We propose a VN allocation model for multi-tenant data centers and formulate a problem that deals with the bandwidth loss in a single VN due a single SN failure. We conduct numerical simulations (with the setting that has 1.7×10^8 bit/s bandwidth demand on each VN, (denoted by C_i)). When each node in each VN is scattered and mapped to an individual physical server, each VN can have the minimum bandwidth loss (5.3×10^2 bit/s ($3.0 \times 10^{-6} \times C_i$)) but the maximum required bandwidth between physical servers (1.0×10^9 bit/s ($5.7 \times C_i$)). The balance between the bandwidth loss and the required physical resources can be optimized by assigning every four nodes of each VN to an individual physical server, meaning that we minimize the bandwidth loss without over-provisioning of core switches.

key words: *data center, multi-tenant, virtual network allocation, multiple simultaneous failures, fault tolerance*

1. Introduction

A data center for the Infrastructure-as-a-Service (IaaS) type of cloud computing serves virtual data center infrastructures for client organizations, i.e., tenants. In order to host not only business-critical applications but also mission-critical ones in a virtual infrastructure, high availability must be ensured through the use of a fault-tolerant design. One of the typical methods for building the virtual infrastructure is to introduce an overlay network architecture based on a tunneling protocol such as VXLAN (Virtual Extensible Local Area Network) [1]. In this architecture, the virtual network (VN) for a tenant is built as an overlay network by connecting VN nodes, i.e., virtual machines (VMs), that are pooled on the physical servers of the physical substrate network (SN) at the data center. Although the topology of the VN is independent of that of the SN, the components of the VN should be appropriately assigned to physical components in the SN in order to share the SN's resources effectively and tolerate SN failures. This paper focuses on clarifying the fault tolerance of the VN in terms of the influence from SN failures and establishing an efficient allocation of resources to the VN. Our goal is to ensure high availability for the VN so that mission critical applications can be hosted on it.

Mapping VNs to the shared SN in the data center faces issues similar to those raised by embedding VNs in a shared ISPs (Internet Service Providers) network. These issues are commonly referred to as the *Virtual Network Embedding* (VNE) problem, which has been a major research topic in network virtualization [2]. In the VNE problem, researchers have proposed to improve the availability, survivability, and resiliency of VNs by minimizing the network disconnections and capacity loss due to physical link failures [3], [4] and by minimizing the sum of all working and backup resources of physical nodes and links [5]–[7]. Similar to VNEs, a number of proposals have been made on reliability-aware resource allocation and redundancy provisioning in data center networks. One is an allocation scheme that aims at minimizing the impact of failures on VNs by spreading the VMs across multiple fault-domains while reducing bandwidth consumption in the core area of the SN [8]. Another is a scheme that considers minimum shared backup resources reserved on physical links and nodes after physical failures [9]–[12]. The latter studies [9]–[12] suppose that the VN allocation has an impact on the SN resource consumption in terms of backup and restore resources after SN failures. These studies, however, do not consider fault tolerance of the VN. Although a fault-tolerance metric has been proposed in [8], failure-recovery characteristics and bandwidth loss during the recovery time are out of its scope.

We therefore focus on the failure-recovery characteristics of a single VN and analyze its bandwidth loss as a fault-tolerance metric. For this purpose, in [13], we first hypothesized that the recovery time of a single VN increases with the failure complexity and explained our procedure for controlling this recovery time. We then proposed a model for multi-tenant data center networks and formulated a problem that deals with the impact of failures in the SN, expressed in terms of bandwidth loss in each VN. In this paper, we describe scale-out scenarios for the SN to account for what occurs in actual data centers. We also detail a heuristic method that solves the problem. We thereby examine how much the VN allocation affects the bandwidth loss on failure. We describe that the bandwidth loss in a single VN is highly dependent on whether components in the VN are consolidated in a few physical components or distributed to many phys-

Manuscript received March 27, 2015.
Manuscript revised June 23, 2015.

[†]The author is with the IT Platform Division Group, Information & Telecommunications Systems Company, Hitachi, Ltd., Tokyo, 103-0027 Japan.
[††]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
a) E-mail: yukio.ogawa.xq@hitachi.com

DOI: 10.1587/transcom.E98.B.2121

ical components. We also show the trade-off between the bandwidth loss and the physical resources, i.e., bandwidth and power, required by the VN in the SN.

The rest of this paper is organized as follows. In Sect. 2, we present our hypothesis about the VN recovery time. In Sect. 3, we describe a model of a multi-tenant network and the problem expressing our allocation scheme. In Sect. 4, we present a heuristic for solving the problem. In Sect. 5, we evaluate the VN allocation, and finally, in Sect. 6, give conclusions.

2. A Hypothesis on the Failure Recovery Time

In virtualized environments, many VNs are consolidated into the SN for better physical resource utilization as well as cost effectiveness. Multiple components of VNs thereby share a single physical component in the SN. As a result, the single failure of, e.g., a single physical server can simultaneously disrupt not only multiple VNs but also multiple nodes and links in a single VN. This characteristic significantly impacts the availability of the VN, as compared to the traditional network composed of dedicated physical components. Previous studies have shown that multiple simultaneous failures in a network can lead to a longer recovery time [14] as a result of, e.g., BGP (Border Gateway Protocol) convergence delays in inter-AS (Autonomous Systems) routing [15], the complexity of fault localization in large enterprise systems [16], and SRG (Shared Risk Group) failures in optical networks [4]. On the basis of our knowledge and experience, we believe that the same problems exist in VNs in a multi-tenant data center.

Based on the above discussion, the hypothesis for the relationship between the failure complexity and the network recovery time is illustrated in Fig. 1. When the complexity of failures in a single VN is low, e.g., when only a few components in the VN fail simultaneously, the VN can recover after a few seconds by automatically switching to hot-standby nodes and links. This can be done by using existing autonomous decentralized control techniques such as VRRP (Virtual Router Redundancy Protocol) [17] and VMware FT (Fault Tolerance) [18][†]. We call this technique *hot-standby recovery*. On the other hand, if the complexity of the failure is high, the VN has a great risk of inducing unexpected behaviors from misconfigurations, software bugs, etc. [19], [20]. This behavior can prevent the failed nodes and links from switching to standby ones. It can thereby significantly delay the restoration of the VN, which may end up taking several minutes or even hours. Since this type of failures depends on the implementation and configuration of the VN, it is difficult to predict the occurrence of such failures as well as their duration in advance. We hence adopt a centralized control technique by which the failed nodes are forced to be terminated and cold-standby nodes are alternatively booted [21], [22]; this technique can reduce the network downtime to a few minutes. We call this technique *cold-standby recovery*.

As explained above, we hypothesize that the complicated failures in a single VN, resulting from multiple logical components failing simultaneously due to a single failure in the SN, can cause the VN to have significantly longer downtime. We thus propose a procedure for reducing this downtime; this procedure combines hot- and cold-standby recovery and switches from the former to the latter with reference to the failure complexity.

3. Virtual Network Allocation

Here, we propose a multi-tenant network model and a VN allocation scheme for tolerating SN failures.

3.1 Network Model in a Multi-Tenant Environment

Figure 2 gives an overview of mapping a VN onto the underlay SN. In what follows, we give models of the VN and the SN, both of which are defined including end nodes, i.e., end VMs and physical servers, respectively.

The SN is modeled with the following sets and parameters.

$G_u = (V, W, E)$: the SN topology consisting of the set of

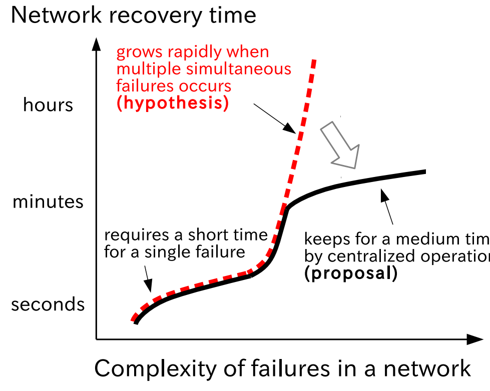


Fig. 1 Hypothesis of time to recovery from failures in a single VN.

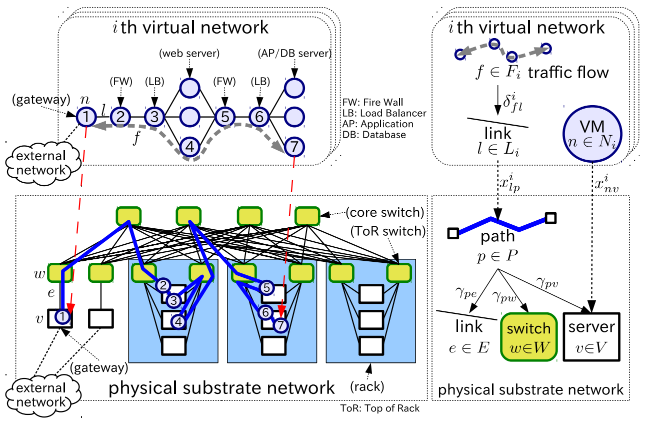


Fig. 2 Network model for a multi-tenant data center.

[†]VMware is a registered trademark or a trademark of VMware, Inc. in the United States and/or other jurisdictions.

end nodes (i.e., physical servers) V, the set of intermediate nodes (i.e., physical switches) W, and the set of physical links E. The identifiers of a physical server, a physical switch, and a physical link are v, w, and e, respectively.

P : the set of physical paths between the pairs of physical servers. The identifier of a physical path is p.

R_v : the constant number of CPU cores in a physical server $v \in V$.

S_e : the constant bandwidth of a physical link $e \in E$.

D_v, D_w, D_e : the constant failure rates (the number of failures per device per unit time) of a physical server $v \in V$, a physical switch $w \in W$, and a physical link $e \in E$, respectively.

Furthermore, each physical path $p \in P$ is mapped onto a physical server $v \in P$, a physical switch $w \in W$, and a physical link $e \in E$ by using the following parameters.

$\gamma_{pv}, \gamma_{pw}, \gamma_{pe} \in \{0, 1\}$: the binary variables that take a value of 1 if each physical path $p \in P$ is mapped onto a physical server $v \in V$, a physical switch $w \in W$, and a physical link $e \in E$, respectively, and 0 otherwise. The values of γ_{pv} and γ_{pw} depend on the value of γ_{pe}.

Note that, though each physical server accesses external storage via a storage area network (SAN), we will omit the SAN because it hardly affects the VN allocation in our model.

Now, let I be the set of VNs hosted on the SN. The ith VN is defined as follows.

$G_o^i = (N_i, L_i)$: the ith VN topology consisting of the set of logical nodes (i.e., VMs) N_i and the set of logical links L_i. The identifiers of a VM and a logical link are n and l, respectively.

F_i : the set of traffic flows between the pairs of end VMs. The identifier of a traffic flow is f.

r_n^i : the constant number of CPU cores required by a VM $n \in N_i$.

c_f^i : the constant average bandwidth of a traffic flow $f \in F_i$. Moreover, the constant total average bandwidth for accessing the services offered on the ith VN from an external network is defined as C_i ($C_i = \sum_{f \in F_i} c_f^i$).

Furthermore, in the ith VN, each traffic flow $f \in F_i$ is assigned to a logical link $l \in L_i$ by using the following parameters.

δ_{fl}^i : the binary variable that takes a value of 1 if a traffic flow $f \in F_i$ is routed through a logical link $l \in L_i$ and 0 otherwise.

In the multi-tenant model, each node $n \in N_i$ and each link $l \in L_i$ in the ith VN are mapped onto a physical server $v \in V$ and a path $p \in P$ in the SN, respectively. They are defined by the following binary variables.

$x_{lp}^i \in \{0, 1\}$: the binary variable that takes a value of 1 when each logical link $l \in L_i$ is mapped onto a physical path $p \in P$ and 0 otherwise.

$x_{nv}^i \in \{0, 1\}$: the binary variable that takes a value of 1 when each logical node $n \in N_i$ is mapped onto a physical server $v \in V$ and 0 otherwise. The value of x_{nv}^i depends on the value of x_{lp}^i.

The the ith VN also has the following attributes.

T_v^i, T_w^i, T_e^i : the recovery time periods of the ith VN after a failure happens on a physical server $v \in V$, a physical switch $w \in W$, and a physical link $e \in E$, respectively. These are explained in Sect. 3.3.

3.2 Problem Description

In this paper, the goal of VN allocation is to minimize the bandwidth loss when failures happen in the SN. The objective function minimizes the total bandwidth loss summed over all VNs. Let B_i denote the bandwidth loss of the ith VN, which is defined as the sum of the losses caused by the failures of physical servers, physical switches and physical links. This optimization problem can be formulated as follows:

Objective: minimize

$$\sum_{i \in I} B_i = \sum_{i \in I} \left(\sum_{v \in V} B_v^i + \sum_{w \in W} B_w^i + \sum_{e \in E} B_e^i \right), \tag{1}$$

where B_v^i, B_w^i, and B_e^i are the bandwidth losses of the ith VN resulting from the failure of a physical server $v \in V$, a physical switch $w \in W$, and a physical link $e \in E$, respectively. B_v^i is defined as the product of the VN's failure rate and the total amount of lost traffic, both of which are caused by the failure of a physical server $v \in V$. B_w^i and B_e^i are defined in the same manner. These variables are therefore formulated as

$$B_v^i = D_v T_v^i \sum_{f \in F_i} X_{fv}^i c_f^i, \tag{2}$$

$$B_w^i = D_w T_w^i \sum_{f \in F_i} X_{fw}^i c_f^i, \tag{3}$$

$$B_e^i = D_e T_e^i \sum_{f \in F_i} X_{fe}^i c_f^i, \tag{4}$$

where X_{fv}^i, X_{fw}^i and X_{fe}^i are the binary variables that respectively indicate the assignments of the ith VN's traffic flow $f \in F_i$ to a physical server $v \in V$, a physical switch $w \in W$, and a physical link $e \in E$. Each of these variables is given below by using the notation $\bigcup$, which means logical sum here.

$$X_{fv}^i = \bigcup_{p \in P} \bigcup_{l \in L_i} \gamma_{pv} x_{lp}^i \delta_{fl}^i, \tag{5}$$

$$X_{fw}^i = \bigcup_{p \in P} \bigcup_{l \in L_i} \gamma_{pw} x_{lp}^i \delta_{fl}^i, \tag{6}$$

$$X_{fe}^i = \bigcup_{p \in P} \bigcup_{l \in L_i} \gamma_{pe} x_{lp}^i \delta_{fl}^i, \tag{7}$$

Note that we do not consider multiple simultaneous failures in the SN because the probability of multiple failures is much smaller than that of a single failure in the SN. We also assume that failures in a single VN do not spread to other VNs.

The constraints on Objective (1) are stated below.

Subject to:

$$\sum_{p\in P} x^i_{lp} = 1, \quad \forall l\in L_i, i\in I \tag{8}$$

$$\sum_{i\in I}\sum_{n\in N_i} x^i_{nv} r^i_n \leq a_1 R_v, \quad \forall v\in V \tag{9}$$

$$\sum_{i\in I}\sum_{p\in P} \gamma_{pe} \sum_{l\in L_i} x^i_{lp} \sum_{f\in F_i} \delta^i_{fl} c^i_f \leq a_2 S_e, \quad \forall e\in E \tag{10}$$

Constraint (8) ensures that each logical link is certainly assigned to a physical path; this assignment implies both end nodes of the link are embedded, too. Constraint (9) ensures that the ratio of the sum of the CPU cores required by the VMs assigned to a physical server to the CPU cores on the physical server is less than a_1. Constraint (10) ensures that the ratio of the bandwidth sum of the traffic flows going through a physical link to the bandwidth provided by the physical link is less than a_2. Here, a_1 satisfies $0 < a_1 < 1$, and it guarantees each physical server provides CPU cores for standby VMs after any single failure in the SN. a_2 also satisfies $0 < a_2 < 1$, and it ensures that each physical link carries fail-over traffic after a single failure in the SN.

3.3 Modeling Recovery Time of a Virtual Network

On the basis of the hypothesis in Sect. 2, we present a model for recovery time periods of the ith VN, i.e., T^i_v, T^i_w, and T^i_e, after the failures of a physical server $v\in V$, a physical switch $w\in W$, and a physical link $e\in E$ in Eqs. (2), (3) and (4).

As explained in Sect. 3.1, each VM and each link in a single VN are mapped onto a physical server and a physical path in the SN, respectively. In this type of data center network, when a failure occurs in a physical switch/link and disrupts the VN links mapped onto the physical switch/link, recovery of the physical switch/link leads to recovery of the VN links. The VN itself does not have a capability of recovering from such failures and relies on the recovery mechanism (e.g., equal-cost multi-path routing) of the SN [23]. On the other hand, if a failure occurs in a physical server and it has an impact on a VN having VMs embedded in the physical server, the VN should recover the VMs by utilizing its own failure-recovery mechanism.

In accordance with the above mechanisms, T^i_w and T^i_e are approximately equal to the recovery time of the physical switch $w\in W$ and the physical link $e\in E$, respectively. T^i_w and T^i_e are thus defined as constants not influenced by how the ith VN is embedded in the physical switch $w\in W$ and the physical link $e\in E$. By contrast, T^i_v depends on how complicated the ith VN becomes as a result of the failure of the physical server $v\in V$. We can express the level of complexity as the number of multiple VMs failing simultaneously in the ith VN as a result of a failure of the physical server $v\in V$; this number is equivalent to the number of VMs in the ith VN that have been assigned to the physical server $v\in V$. Let m^i_v denote this number. As explained in Sect. 2, the ith VN recovers from the failure by performing the procedure below.

- Each of the VMs in the ith VN is paired with a dedicated hot-standby VM in advance. When a VM fails, the paired hot-standby VM takes over in the case of $m^i_v \leq \theta$, where θ is a threshold parameter.
- Redundant shared resources are set aside for cold-standby VMs in each physical server. When a VM in the ith VN fails, an alternative cold-standby VM is booted to succeed it in the case of $m^i_v > \theta$.

T^i_v is thus modeled as follows. In general, the service time distribution commonly used in computer systems is an exponential distribution (with mean μ and variance μ^2) [24]. We assume that each VM's processing time to recover from a physical server failure also follows this distribution. T^i_v is defined as the maximum recovery time among m^i_v VMs; this recovery time is approximated as the sum of the mean and the standard deviation time among m^i_v VMs. We assume that m^i_v VMs fail coincidentally and recover independently of each other. The expected standard deviation of the m^i_v VMs' recovery delays is thus $\sqrt{\frac{m^i_v-1}{m^i_v}}\,\mu$. We define that the m^i_v VMs recover after a mean time μ_h through hot-standby recovery in the case of $m^i_v \leq \theta$, or after a mean time μ_c through cold-standby recovery otherwise. T^i_v is consequently defined as (see Fig. 3).

$$T^i_v = \begin{cases} \left(1 + \sqrt{\frac{m^i_v-1}{m^i_v}}\right)\mu_h & (m^i_v \leq \theta) \\ \left(1 + \sqrt{\frac{m^i_v-1}{m^i_v}}\right)\mu_c & (m^i_v > \theta) \end{cases} \tag{11}$$

As explained above, if more than θ VMs in the VN are assigned to a physical server, these VMs will recover through cold-standby recovery. Because the VN has to prioritize its fault tolerance and needs to shorten the recovery time of the VMs, the following constraint is added to those in Sect. 3.2 and used in the evaluations in Sect. 5.

Subject to:

$$\sum_{n\in N_i} x^i_{nv} \leq \theta, \quad \forall i\in I, \forall v\in V \tag{12}$$

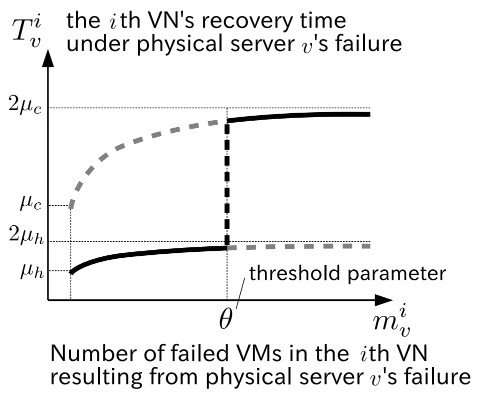


Fig. 3 Recovery time model of a single VN.

Constraint (12) prohibits the VN from assigning more than θ VMs to a physical server, thereby ensuring that the VMs will recover only through hot-standby recovery.

3.4 Scale-Out Scenarios for Physical Network

To account for what occurs in actual data centers, the SN starts with a minimum configuration, and a physical component is then added to the SN when the resources provided by the components in the SN become insufficient. The SN should thus have a scale-out architecture. As shown in lower left of Fig. 2, let's suppose the SN is mostly composed of core switches and racks. Moreover, each rack includes Top-of-Rack (ToR) switches and physical servers. We have following three cases for scaling out the SN.

1. A rack is added to the SN when either Constraint (9) or Constraint (10) fails to hold in all the current racks in the SN.
2. A core switch is append to the SN when Constraint (10) fails in the core area of the SN.
3. Either a rack or a core switch is added to the SN if Constraint (12) is violated.

Here, all of the physical links among a newly added device and existing components are connected at the time of adding the newly device. Note that, although physical servers connected to an external network (called gateway servers) and the link connecting the gateway servers (see Fig. 2) are also scaled out, we will omit their explanations here for simplicity.

When the VNs are allocated in a fault-tolerant manner, the SN is scaled out in all the cases. In particular, the third case adds a physical component in order to minimize the failure recovery time of a single VN, even if there are unused available resources in the SN. In addition, we provide another scale-out scenario. Here, when the VNs are allocated in a resource-oriented manner, the SN expands by checking the first and the second cases, but not the third case. This sort of operation is the usual practice in actual data centers, and we will evaluate it for comparison purposes.

4. An Evaluation Method for Virtual Network Allocation

The VN allocation problem explained in Sect. 3.2 is a sort of VN embedding problem [2], which reduces to an multi-way separator problem that is *NP*-hard to solve optimally [25]. We thus apply a method based on well-established heuristics; this is used as a simple and feasible algorithm for finding a near optimal solution of such a problem. The method is composed of a two-stage allocation procedure. Upon receiving a tenant system request, a VN is initially allocated according to the *Greedy Algorithm* [26]. The initial allocation is then refined by the *Tabu search* [27].

The first stage utilizes the fact that a gateway server allocates its resources only to a VM connected to the external network (called a gateway VM) (see Fig. 2). We thereby begin by assigning gateway VMs to corresponding gateway servers, as explained in Algorithm 1. A link connecting an assigned VM and an unassigned VM is then iteratively mapped so as to minimize Objective (1), until all of the VMs and the links in the VN are assigned. In this stage, the SN scales out according to the scenario described in Sect. 3.4. The second stage, as explained in Algorithm 2, repeatedly moves each of the assigned VMs in the VN to another physical server to find a better assignment in the neighborhood of the current placement, until all possible assignments are checked or the number of VM replacements is above a threshold.

Algorithm 1 Initial allocation of a single VN

```
1: Assign gateway VMs to corresponding gateway servers.
2: for each logical link $l$ connecting an assigned VM $n$ and an unassigned
   VM do
3:     Select the physical paths starting at the physical server to which the
       VM $n$ is assigned and satisfying the constraints.
4:     if not selected then
5:         if the SN has attained its maximum configuration then
6:             Stop the allocation of the VN.
7:         else
8:             Scale out the SN according to Sec. 3.4.
9:         end if
10:    end if
11:    Sort the physical paths in ascending order by each path attributes:
       hop count, available CPU cores of the end physical server, and avail-
       able bandwidth of the bottleneck link.
12:    for each physical path $p$ of the physical paths do
13:        Tentatively assign the logical link $l$ to the physical path $p$ and
           calculate Objective (1).
14:        if Objective (1) reaches the minimum value then
15:            Update the assignment.
16:        end if
17:    end for
18: end for
```

These two algorithms try to assign a shorter physical path to a logical link in the VN in order to reduce the bandwidth consumed in the core of the SN. Note that the choice of above method is not the main focus in this paper and we thus omit to show the effectiveness of the method itself in the next section.

5. Evaluation

Here, we describe the trade-off between the fault tolerance and the SN resource usage when VNs are mapped to the SN in the fault-tolerant manner.

5.1 Multi-Tenant Data Center Network for Evaluation

Each VN was configured to have an active-active topology for mission-critical applications; half of this VN topology is shown in the upper left of Fig. 2. Each node in half of the VN was paired with a dedicated node in the other half for the purpose of hot-standby recovery. Corresponding to the VN topology, the SN components (i.e., gateway servers, core switches, and ToR switches and physical servers in each

Algorithm 2 Optimization of a single VN allocation

```
1: Provide a tabu list (= {}).
2: while iteration counts < a threshold do
3:   Provide a list for registering neighboring allocations (called a neighbor list) (= {}).
4:   for each VM $n$ (mapped to a physical server $v$) of the current VN allocation do
5:     Select the physical servers neighboring the server $v$ so as not to increase the total physical path length.
6:     for each physical server $v'$ of the physical servers do
7:       Change the target of the VM $n$ from $v$ to $v'$.
8:       if the changed VN allocation is not in the tabu list and satisfies the constraints then
9:         Input the changed allocation to the neighbor list.
10:      end if
11:    end for
12:  end for
13:  if there are no allocations in the neighbor list then
14:    Stop the allocation of the VN.
15:  end if
16:  Calculate Objective (1) for each allocation in the neighbor list and select the allocation which minimizes Objective (1) (called the best candidate).
17:  if Objective (1) of the best candidate is less than that of the best, i.e., optimized, allocation then
18:    Update the best allocation with the best candidate.
19:  end if
20:  Update the current allocation with the best candidate.
21:  Input the best candidate to the tabu list.
22: end while
```

Table 1 Energy properties of SN components.

	power (W)	EPI (%) [a]	
	max	max	min
ToR & core switch	165 [29]	40 [30]	8 [31]
physical server	750 [32]	80 [33]	40 [33]

[a] Energy Proportionality Index [28] (also called Dynamic Range in [33])

rack shown in lower left of Fig. 2) were divided into two symmetrical parts. Both halves of the SN allocated their resources to corresponding halves of the VN. As both allocations were exactly the same, we will only explain half of the allocations.

The SN had a two-level fat-tree topology configured as an non-oversubscribed network (see Fig. 2). The SN used a 32-port switch for both the ToR and core switches; this SN thereby consisted of 8 core switches, 16 ToR switches, and 120 physical servers in its maximum configuration. Each rack included a ToR switch and 8 physical servers. The number of CPU cores in each physical server, R_v, was 32, and the bandwidth of each link, S_e, was 1.0×10^{10} bit/s. Furthermore, energy properties of each physical switch and each physical server are defined in Table 1. Here, EPI is a metric for quantifying energy proportionality of physical devices [28]. In addition, a_1 in Constraint (9) and a_2 in Constraint (10) were set to 0.9 and 0.5, respectively; these values were determined by considering fail-over after any single physical failure. Under the above settings at the maximum configuration, 3,360 CPU cores were made available for allocation.

Table 2 Failure rates (in failures per year) of SN components.

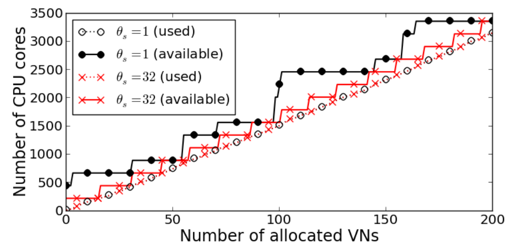
D_v	2.53 [36], 3, 6 [37]
D_w	0.005 - 0.111 [19], 0.055 - 0.073 [35]
D_e	0.054, 0.095 [19], 0.073 [37]

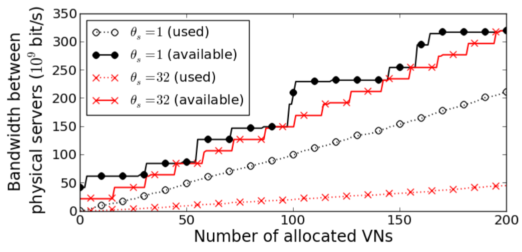
The VN was modeled after the traditional three-tier web serving architecture [34] illustrated in the upper left of Fig. 2. To make the VN model simple, the numbers of web servers and application (AP)/database (DB) servers in the VN were set to the same value; each number followed a truncated normal distribution with mean 5, standard deviation 3 and lower limit 2. Under these settings, each VN had 5.8 web and AP/DB servers and a total of 15.7 VMs (except for the gateway VM) on average. Each traffic flow was defined per path routed through a pair of web and AP/DB servers and had an average bandwidth of 3.0×10^7 bit/s. The average bandwidth for accessing the services offered by these servers from an external network, C_i, was thereby 1.7×10^8 bit/s. Moreover, the number of CPU cores required by each VM, r^i_n, was set to 1. The hot-standby recovery time of each VM, μ_h, was set to 4 s, and the cold-standby recovery time, μ_c, was set to 60 s. Note that, although the number of web servers and that of AP/DB servers in a single VN could be different and the VN could have various topologies, no qualitative difference existed in the results explained in the next subsections. We also note that the values of C_i and other parameters were based on our knowledge and experience. These values did not impact the qualitative results in the next subsections either.

There have been several studies on network failures [19], [35]–[37]. Table 2 summarizes the failure rate (in failures per device per year) of a single physical server, D_v, that of a single physical switch, D_w, and that of a single physical link, D_e, and their source references. Note that the higher failure rate of D_v is due to software-related errors of the operating systems and hypervisors [36], [37]. In our evaluations, D_v was set to 4, and D_w and D_e were neglected because they are much smaller than D_v. Objective (1) thus depended on the first term alone.

5.2 Evaluation Results

To evaluate Objective (1), the fault-recovery time is given by Eq. (11), in which the threshold θ must take various values from 1 to 32 depending on many factors such as processes, configurations, software implementations of VMs in each VN. Here, the maximum value, 32, was set to be equal to R_v/r^i_n. In actual operations, it is difficult to define the actual value of θ in advance. We therefore initially chose an assumed value of θ, θ_s, and allocated VNs so as to minimize Objective (1) by using θ_s. We then evaluated the allocations when the actual value of θ took various ones. Note that horizontal positions of the plots in this section have been adjusted to keep the error bars visible. In Figs. 5, 6 and 7, each marker specifies the mean and each error bar specifies the 5% and 95% values for all VN allocations.


(a) number of CPU cores


(b) Bandwidth between physical servers

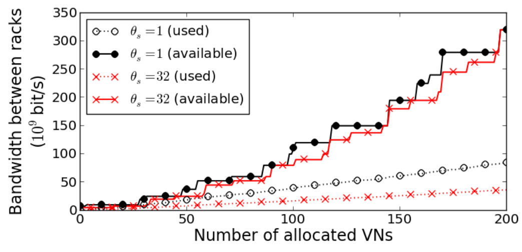
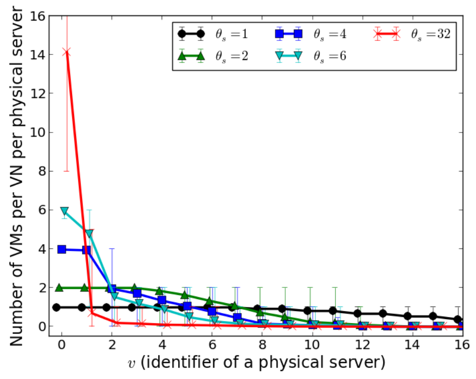

(c) Bandwidth between racks

Fig. 4 Changes in physical resources along with VN allocations.


(a) Assignment of VMs

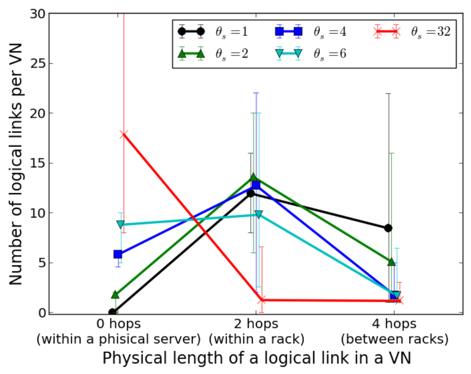

(b) Assignment of links

Fig. 5 VN assignment.

5.2.1 Changes in Physical Resources Along with VN Allocations

Figure 4 shows how the available and used physical resources (i.e., the total number of CPU cores, the total bandwidth between physical servers and that between racks) changed when VNs were allocated with $\theta_s = 1$ or $\theta_s = 32$. While the number of used CPU cores increased linearly with increasing VN allocations, that of available CPU cores increased stepwise with the addition of a single rack or multiple racks to the SN. This was due to Constraint (12), and a smaller θ_s resulted in adding more racks at a time. Under our evaluation settings, a lack of CPU resources triggered adding a new rack to the SN. The increase in available bandwidth between physical servers (i.e., the total bandwidth of physical server links connected to ToR switches) depended on this rack addition and thereby coincided with the increase in available CPU cores. Moreover, the unused bandwidth between racks increased every time a core switch was added, since all the links between the core switch and all the ToR switches were equipped and that provided much more bandwidth than the VNs' demand. Both the consumed bandwidth between physical servers and that between racks depended on θ_s; this will be explained in the next subsection.

5.2.2 Overview of a Single VN Mapping

Figure 5 shows the overview for the VN assignment, where the identifiers on the horizontal axis are sorted in descending order. When θ_s was set to 32, almost all of the VMs in a single VN, except for the gateway VM, were consolidated in a single physical server. Most of the links in the VN were virtually assigned within the physical server and did not occupy the bandwidth of the physical links. In contrast, when θ_s was set to 1, each VM was mapped onto an individual physical server, and each link was mapped onto a physical link between two physical servers. As θ_s became smaller, the VMs became distributed to more physical servers due to Constraint (12), and thereby, more bandwidth of the physical links became occupied. The recovery time T_v^i was assumed to increase drastically due to simultaneous failures of more than θ_s VMs. This resulted in mapping θ_s or less VMs in the VN onto a single physical server. As explained above, the value of θ_s determines the *shape* of the VN; i.e., it determines whether the VN is one with VMs and logical links scattered across many physical servers and physical links, or one consolidating all VMs and links in a few physical servers.

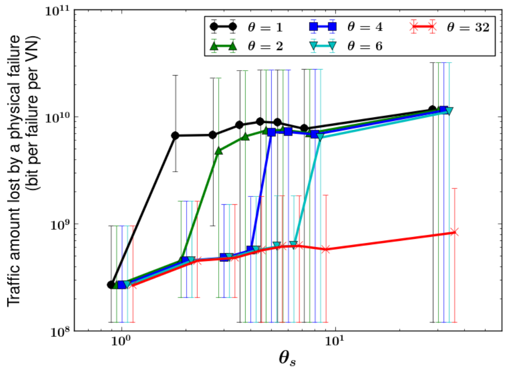
5.2.3 Trade-Off between Fault Tolerance and Physical Resources Consumption

(1) Fault Tolerance

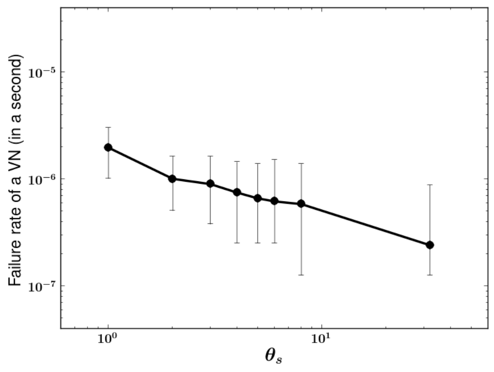
In order to evaluate the relationship between the shape and fault tolerance of a single VN, we analyzed how the optimized Objective (1) changed with θ_s, as well as θ. Figure 6(a) shows the average traffic amount in the VN lost by a physical failure (bit per failure per VN), which corresponds to $T^i_v \sum_{f\in F_i} X^i_{fv} c^i_f$ in Eq. (2). If θ_s was set to 1, the traffic amount was the smallest (2.7×10^8 bit) for any θ. Because the VMs were each distributed to an individual physical server, the traffic flows spread across many physical servers so as to minimize the traffic amount lost by one failure of a physical server. As θ_s increased, the traffic amount increased, as a result of consolidating more VMs and thereby flowing more traffic into a single physical server. Although the traffic amount lost by a failure was also smaller (less than 10^9 bit) for $\theta \geq \theta_s$ at that time, it increased significantly (around 10^{10} bit) for $\theta < \theta_s$, resulting from cold-standby recovery. When θ_s was set to the maximum value, 32, the traffic amount reached the maximum for any θ, except for $\theta = \theta_s$.

Figure 6(b) shows the average failure rate of a single VN, which corresponds to $\sum_{v\in V} D_v \sum_{f\in F_i} X^i_{fv}$ related to Eq. (2). This failure rate decreased from 2.0×10^{-6} to 2.4×10^{-7}, when θ_s was set to a large value and more VMs and traffic flows in a single VN were concentrated in fewer physical servers.

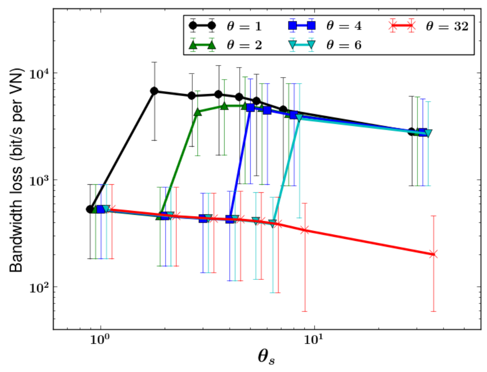
The average bandwidth lost by a physical failure in a single VN, B_i in Objective (1), (Fig. 6(c)) was affected by both the traffic amount lost by a physical failure and the failure rate. When θ_s was set to 1 and each VM was distributed to an individual physical server, the average B_i was nearly the minimum (5.3×10^2 bit/s per VN ($3.0 \times 10^{-6} \times C_i$)) for any θ. This was because each VM used hot-standby recovery even though the failure rate of the VN was high. When θ_s became large and VMs were consolidated in fewer physical servers, the average B_i slightly decreased, as long as $\theta \geq \theta_s$. This was because the decrease in the VN failure rate had more influence than the increase in the traffic amount lost by a physical failure. However, if θ was smaller than θ_s, the average B_i significantly increased to around 6.0×10^3 bit/s per VN ($3.4{\times}10^{-5}{\times}C_i$) because each VM used cold-standby recovery. When θ_s was set to a large value like 32, all the VMs in the VN were concentrated in a single physical server and the failure rate of the VN decreased. In this case, cold-standby recovery was applied unless $\theta \geq \theta_s$. As a result, the average B_i reached the maximum value for any θ other than $\theta \geq \theta_s$



(a) Traffic amount lost by a physical failure



(b) Failure rate



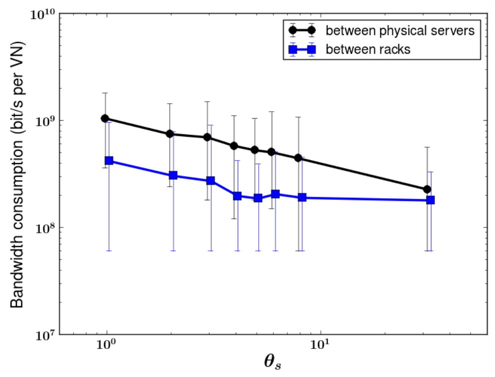
(c) Loss of bandwidth

Fig. 6 Fault tolerance of each VN.

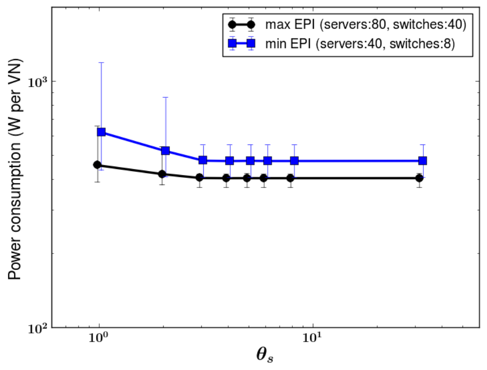
(2) Physical Resources Consumption

In order to describe the influence of the VN's shape on the requirements for the SN, we analyzed how θ_s changed the bandwidth used by each VN outside the physical servers (Fig. 7(a)) as well as the total power consumed by each VN (Fig. 7(b)).

When θ_s was set to 1 and most of the logical links in a single VN were mapped to physical links between and within racks, both the average consumed bandwidth between servers and that between racks reached the maximum (between servers: 1.0×10^9 bit/s per VN ($5.7 \times C_i$), between racks: 4.2×10^8 bit/s per VN ($2.4 \times C_i$)). As θ_s increased and more logical links were consolidated in a physical server,



(a) Physical bandwidth



(b) Power

Fig. 7 Physical resources consumed by each VN.

both bandwidths became smaller. The smallest bandwidth between racks (about 2.0×10^8 bit/s per VN ($1.1 \times C_i$)) was when θ_s was 4; here, the VN had almost no inter-rack traffic flows other than the one coming through the gateway. In this case, almost all of the logical links were mapped onto the physical links between the physical servers and ToR switches. The traffic flows from a VM went to and back from the ToR switch in a rack and were not forwarded outside the rack. Moreover, when θ_s was set to the maximum, 32, all of the logical links except for the one connected to the gateway were embedded in a few physical servers; this minimized both bandwidths.

Furthermore, the average power consumed by the physical devices (i.e., the core and ToR switches and the physical servers in the SN) per VN is shown in Fig. 7(b). Overall, the power consumption of each VN depended on that consumed by physical servers, and the portion of that consumed by core and ToR switches was about 5% in our evaluation settings. If physical servers had achieved perfect energy proportionality, i.e., EPI=100%, average power consumption of each VN (denoted by H_i) should be about 3.7×10^2 W per VN (= (750 W per physical server) × (15.7 VMs per VN) / (32 VMs per physical server)).

While the sum of CPU cores required by the VMs in a single VN was a constant for the VN and did not depend on θ_s, the total number of physical servers required by the VN was influenced by θ_s. When θ_s increased, less physical servers were needed for each VN, as explained in Sect. 5.2.2. The power consumption of each VN denoted the maximum (6.2×10^2 W per VN ($1.7 \times H_i$)) at $\theta_s = 1$ when the EPIs had the minimum values; this was because the small θ_s resulted in spreading the VMs across many non-energy-proportional servers having plenty of unused CPU cores. The power consumption got smaller along with the growth of θ_s and achieved constant values (minimum EPI: 4.8×10^2 W per VN ($1.3 \times H_i$), maximum EPI: 4.1×10^2 W per VN ($1.1 \times H_i$)) at $\theta_s \geq 3$. In this case, only a single rack came to be added to the SN almost every time when the SN was scaled-out. Moreover, the difference between the power consumption at $\theta_s = 1$ and that at $\theta_s = 33$ was not so large (less than 23%) even if the physical components were not energy proportional, because the SN's scale-out architecture increased energy proportionality at the system level.

5.2.4 Results for Another Scale-Out Scenario

As explained in Sect. 3.4, the SN could have another scale-out scenario, called resource-oriented allocation. We evaluated this case with the same settings as described in the previous subsections. Here, a new rack was added only when the available CPU resources had been used up; the value of θ_s did not affect the allocation. As a result, more than θ_s VMs were mapped onto a single physical server, and this caused cold-standby recovery. This made B_i larger than that of the fault-tolerant allocation when θ_s was less than 4. In addition, the resource-oriented allocation always assigned a logical link to a physical link within a rack and added a single rack in scaling-out the SN. These minimized the bandwidth consumed between racks and the power consumption of VNs for any θ_s.

5.2.5 VN Allocation Policy Derived from the Results

As shown in Figs. 6(c) and 7, the risk of bandwidth loss in each VN caused by a physical failure increases with θ_s and the physical resources consumed by each VN, especially the bandwidth consumed by the VN's usual traffic flows, decreases with θ_s. We should consider which θ_s is applicable to actual operations. Although we must reduce the risk of significant service disruptions caused by multiple simultaneous failures in the VN, the excess capacity required for fault tolerance should be kept as low as possible. One of the best approaches is therefore to minimize the bandwidth loss of the VN resulting from sharing physical resources while avoiding holding too many redundant core switches and increasing power consumption. This state is called Pareto optimality [38]. Under our evaluation settings, this was achieved when θ_s was 4. This value maximized the utilization within racks as well as minimized the bandwidth used between racks and the power consumption of each VN. It remained the same for the fault-tolerant allocation and the resource-oriented allocation.

6. Conclusion

We described the fault tolerance of each VN in an IaaS type of data center, focusing on the situation of multiple simultaneous failures in each VN caused by a single physical failure. Through numerical evaluations based on our hypothesis about the VN recovery time, we found the following results. We set C_i, the average bandwidth of 1.7×10^8 bit/s for accessing the services offered on each VN, in advance. We also determined that H_i, the ideal average power consumption of each VN, was about 3.7×10^2 W per VN. When each of the VMs in the VN was mapped to an individual physical server, the bandwidth loss was close to the minimum, 5.3×10^2 bit/s per VN ($3.0 \times 10^{-6} \times C_i$), but the required bandwidth between physical servers and the power consumed by each VN had the maximum, 1.0×10^9 bit/s per VN ($5.7 \times C_i$) and 6.2×10^2 W per VN ($1.7 \times H_i$), respectively. The trade-off between the bandwidth loss and the required physical resources was balanced by assigning every four VMs in the VN to a individual physical server, by which the required bandwidth of the outside racks and the power consumption of each VN were minimized (about 2.0×10^8 bit/s per VN ($1.1 \times C_i$) and 4.1×10^2 W per VN ($1.1 \times H_i$) at the the maximum EPI, respectively). This solution is coincident with a one-rack type of product offering for data centers; this product is delivered as a pre-configured single rack or multiple racks including physical servers, network switches, and virtualization software (e.g., [39]). This paper dealt with a single data center alone. The resource cost and performance would be different in an environment of multiple data centers and wide-area networks (WANs). In the future, we would therefore like to investigate VN allocation over WANs.

References

[1] M. Mahalingam, D.G. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "VXLAN: A framework for overlaying virtualized layer 2 networks over layer 3 networks," http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-00, Aug. 2011.

[2] A. Fischer, J.F. Botero, M.T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," IEEE Commun. Surv. Tutorials, vol.15, no.4, pp.1888–1906, Feb. 2013.

[3] M.R. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," IEEE Trans. Netw. Serv. Manage., vol.10, no.2, pp.105–118, June 2013.

[4] C. Meixner, F. Dikbiyik, M. Tornatore, C. Chuah, and B. Mukherjee, "Disaster-resilient virtual-network mapping and adaptation in optical networks," Proc. ONDM 2013, pp.107–112, April 2013.

[5] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," Proc. 2011 IEEE International Conference on Communications (ICC), pp.1–6, June 2011.

[6] M.F. Habib, M. Tornatore, M. De Leenheer, F. Dikbiyik, and B. Mukherjee, "Design of disaster-resilient optical datacenter networks," J. Lightwave. Technol., vol.30, no.16, pp.2563–2573, Aug. 2012.

[7] A. Jarray and A. Karmouch, "Cost-efficient mapping for fault-tolerant virtual networks," IEEE Trans. Comput., vol.64, no.3, pp.668–681, March 2015.

[8] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D.A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," SIGCOMM Comput. Commun. Rev., vol.42, no.4, pp.431–442, Aug. 2012.

[9] W.-L. Yeow, C. Westphal, and U. Kozat, "Designing and embedding reliable virtual infrastructures," Proc. Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures — VISA'10, pp.33–40, Sept. 2010.

[10] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," Proc. 2012 IEEE Fifth International Conference on Cloud Computing, pp.196–203, June 2012.

[11] M.G. Rabbani, M.F. Zhani, and R. Boutaba, "On achieving high survivability in virtualized data centers," IEICE Trans. Commun., vol.E97-B, no.1, pp.10–18, Jan. 2014.

[12] Q. Zhang, M.F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," Proc. IEEE INFOCOM 2014 — IEEE Conference on Computer Communications, pp.289–297, April 2014.

[13] Y. Ogawa, G. Hasegawa, and M. Murata, "Virtual network allocation for fault tolerance with bandwidth efficiency in a multi-tenant data center," Proc. 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, pp.555–562, Dec. 2014.

[14] G. Hasegawa, T. Horie, and M. Murata, "Proactive recovery from multiple failures utilizing overlay networking technique," Telecommun. Syst., pp.1001–1019, Feb. 2011.

[15] A. Sahoo, K. Kant, and P. Mohapatra, "BGP convergence delay after multiple simultaneous router failures: Characterization and solutions," Comput. Commun., vol.32, no.7-10, pp.1207–1218, May 2009.

[16] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D.A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," SIGCOMM Comput. Commun. Rev., vol.37, no.4, pp.13–24, Aug. 2007.

[17] R. Hinden, "Virtual router redundancy protocol (VRRP)." http://tools.ietf.org/html/rfc3768, Oct. 2012.

[18] VMware, Inc., "Protecting mission-critical workloads with VMware fault tolerance." http://www.vmware.com/files/pdf/resources/ft_virtualization_wp.pdf, Feb. 2009.

[19] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers," SIGCOMM Comput. Commun. Rev., vol.41, no.4, pp.350–361, Aug. 2011.

[20] R. Potharaju and N. Jain, "When the network crumbles: An empirical study of cloud network failures and their impact on services," Proc. 4th Annual Symposium on Cloud Computing — SOCC'13, pp.1–17, Oct. 2013.

[21] X. Wu, D. Turner, C.-C. Chen, D.A. Maltz, X. Yang, L. Yuan, and M. Zhang, "NetPilot: Automating datacenter network failure mitigation," SIGCOMM Comput. Commun. Rev., vol.42, no.4, pp.419–430, Aug. 2012.

[22] VMware, Inc., "VMware vSphere high availability," http://www.vmware.com/products/datacenter-virtualization/vsphere/high-availability.html, Oct. 2012.

[23] M.F. Bari, R. Boutaba, R. Esteves, L.Z. Granville, M. Podlesny, M.G. Rabbani, Q. Zhang, and M.F. Zhani, "Data center network virtualization: A survey," IEEE Commun. Surv. Tutorials, vol.15, no.2, pp.909–928, 2013.

[24] R. Jain, The Art Of Computer Systems Performance Analysis, John Wiley & Sons, 1991.

[25] D.G. Andersen, "Theoretical approaches to node assignment," Computer Science Department, pp.1–12, Dec. 2002. http://repository.cmu.edu/compsci/86

[26] T.H. Cormen, C. Stein, R.L. Rivest, and C.E. Leiserson, Introduction to Algorithms, 2nd ed., McGraw-Hill Higher Education, 2001.

[27] S. Luke, Essentials of Metaheuristics, Lulu, 2009. Available at http://cs.gmu.edu/~sean/book/metaheuristics/

[28] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," Proc. NETWORKING 2009, Lecture Notes in Computer Science, vol.5550, pp.795–808, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[29] Hitachi Metals, Ltd., "APRESIA," http://www.apresia.jp/en/products/ent/series/10g_1g.html, Dec. 2014.

[30] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "CARPO: Correlation-aware power optimization in data center networks," 2012 Proc. IEEE INFOCOM, pp.1125–1133, March 2012.

[31] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," Proc. USENIX NSDI 2010, p.17, April 2010.

[32] Dell Inc., "PowerEdge R820 rack server details," http://www.dell.com/us/business/p/poweredge-r820/pd, Dec. 2014.

[33] D. Wong and M. Annavaram, "KnightShift: Scaling the energy proportionality wall through server-level heterogeneity," Proc. 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, pp.119–130, Dec. 2012.

[34] I. Pepelnjak, "Introduction to virtualized networking," http://www.ipspace.net/Introduction_to_Virtualized_Networking, Jan. 2012.

[35] W.E. Smith, K.S. Trivedi, L.A. Tomek, and J. Ackaret, "Availability analysis of blade server systems," Ibm. Syst. J., vol.47, no.4, pp.621–640, Oct. 2008.

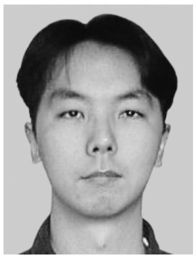
[36] L.A. Barroso and U. Hōlzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 1st ed., Morgan and Claypool Publishers, 2009.

[37] D.S. Kim, F. Machida, and K.S. Trivedi, "Availability modeling and analysis of a virtualized system," Proc. 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing, pp.365–371, Nov. 2009.

[38] K. Miettinen, "Nonlinear multiobjective optimization," International Series in Operations Research & Management Science, vol.12, Springer US, Boston, MA, 1998.

[39] Hitachi Data Systems, "Hitachi unified compute platform solution (UCP) family," http://www.hds.com/, May 2014.

Yukio Ogawa received his M.S. degree in Science from Nagoya University, Japan, in 1994, and Ph.D. degree in Information Science and Technology from Osaka University, Japan, in 2012. He joined the Hitachi Central Research Laboratory in Japan, in 1994. He is currently a senior engineer in IT Platform Division Group, Hitachi, Ltd. in Japan. His research interests include network architectures for cloud systems. He is a member of IEEE and IEICE.



Go Hasegawa received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Osaka, Japan, in 1997 and 2000, respectively. From July 1997 to June 2000, he was a Research Assistant in the Graduate School of Economics, Osaka University. He is now an Associate Professor at the Cybermedia Center, Osaka University. His research is in the area of transport architecture for future high-speed networks and overlay networks. He is a member of IEEE and IEICE.

Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Science from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined the Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with the Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. In April 1999, he became a Professor at the Cybermedia Center, Osaka University, and since April 2004, he has been with the Graduate School of Information Science and Technology, Osaka University. He has had more than five hundred papers published in international and domestic journals and conferences. His research interests include computer communication network architecture, performance modeling, and evaluation. He is a member of IEEE, ACM, and IEICE. He became chair of the IEEE COMSOC Japan Chapter in 2009. He is also working at the National Institute of Information and Communications Technology (NICT) as Deputy of the New-Generation Network R&D Strategic Headquarters.