

Unavailability-aware Shared Virtual Backup Allocation Model for Middleboxes

Fujun He and Eiji Oki

Graduate School of Informatics, Kyoto University, Kyoto, Japan

Abstract—Network function virtualization paradigm enables us to implement network functions provided in middleboxes as softwares which run on commodity servers. This paper proposes an unavailability-aware backup allocation model with shared protection for middleboxes with comprehensively considering the failure, repair, and recovery behaviors of functions and backup servers. Multiple functions can share the backup resources on the backup server. The proposed model aims to find the assignment of backup servers to functions to minimize the maximum unavailability among functions. The multiple situations of failure, repair, and recovery of functions and backup servers lead to several different states for each function. The unavailability of function is estimated through analyzing all states that a function can be in. To compute the unavailability of middlebox for a given backup allocation, an analytical approach is developed based on the queueing theory. With the analytical approach, we introduce a simulated annealing heuristic to solve the backup allocation problem. The results reveal that, compared to a baseline model, the proposed unavailability-aware model reduces the maximum unavailability 11% in average in our examined scenarios.

Index Terms—Network function virtualization, backup allocation, unavailability, queueing theory

I. INTRODUCTION

Middleboxes provide network functions, such as firewalls, intrusion detection systems (IDS), network address translators (NAT), and load balancers (LD), for network services in datacenters and enterprise networks. A traditional middlebox is implemented as a dedicated hardware device providing one type of network function. With adopting network function virtualization (NFV) technology [1], middleboxes can be implemented as softwares running on commodity servers to efficiently utilize the hardware resources in networks.

The unavailability of middlebox is a critical issue. Middlebox failures frequently occur in networks, which significantly degrade the network performance [2]. If a middlebox fails and cannot be promptly recovered, the network function becomes unavailable, which interrupts all services utilizing the function. One approach to reduce the unavailability of a middlebox is to prepare a standby in a backup server, where each internal state of the active middlebox is synchronized to its standby [3]–[6]. Once the failing of an active middlebox is identified, if its backup server does not fail, all traffic handled by the middlebox is immediately diverted to the standby, which recovers the functionality of failed middlebox. This approach provides a prompt recovery with a high probability that the protection successes. However, this straightforward approach requires a large amount of computing resources.

Instead of utilizing the dedicated computing resources to entirely run a standby for each function, a more flexible approach

leverages the flexibility of NFV technology with allowing resource sharing among standbys of different functions [7]–[10]. In this approach, snapshots of multiple functions are stored and updated to the same backup server. When a protected function fails, it can be recovered on the backup server, where some missing states are regenerated by the stored snapshot [11]. The backup server may not recover all failed functions at the same time due to the limited capacity. Since updating the stored snapshots of functions usually utilizes much less resources compared to running or recovering the same number of functions at the same time, this approach with shared protection can significantly reduce the required backup resources. Note that a backup server in this paper can be a logical server abstracted from a set of resources and is not necessarily referred to a physical one.

In literature, the backup allocation problem with the shared protection has been studied from different aspects. The work in [7] considered finding the assignment of backup servers to functions with maximizing the probability for a recovery of all failed functions. The work in [8] studied the maximum number of failed functions that can be fully recovered by a given number of backup servers. The works in [9], [10] presented a probabilistic protection model to minimize the required backup capacity, where the probability that the protection fails is guaranteed within a given number.

The unavailability of middlebox is not always explicitly considered, such as in the works of [7]–[10]. As a result, the suggested backup allocations may not be optimal in terms of the middlebox unavailability. Handling the middlebox unavailability directly in a backup allocation model requires comprehensive analyses from several aspects; the work in [4] studied it with considering the dedicated protection. It becomes difficult when the shared protection is adopted. Basically, there are two questions: 1) how can we estimate the unavailability of middlebox with considering the shared protection? 2) what is the optimal backup allocation in terms of the unavailability of middlebox? The two questions have not been addressed and are studied in this paper.

This paper proposes an unavailability-aware backup allocation model with the shared protection for middleboxes with comprehensively considering the failure, repair, and recovery behaviors of functions and backup servers. In our model, multiple functions can share the backup resources on the backup server. The proposed model aims to find the assignment of backup servers to functions to minimize the maximum unavailability among functions. The multiple situations of failure, repair, and recovery of functions and backup servers

lead to several different states for each function. The unavailability of function is estimated through analyzing all states that a function can be in. We develop an analytical approach based on the queueing theory to compute the unavailability of middlebox for a given backup allocation. Based on the analytical approach, a simulated annealing (SA) heuristic is introduced to solve the backup allocation problem. The performance dependencies on the failure, repair, and recovery behaviors of functions and backup servers are evaluated. We compare the proposed model with a baseline model. The results reveal that the proposed unavailability-aware model reduces the maximum unavailability 11% in average compared to the baseline model in our examined scenarios.

II. RELATED WORK

In literature, several works studied the dedicated protection to provide a prompt recovery with a high probability to succeed [3]–[6]. The work in [3] presented a framework with rollback recovery for middleboxes, where protection overhead is reduced by introducing ordered logging and parallel release. The work in [4] considered the heterogeneous failure processes to minimize the required backup capacity with satisfying the availability requirements. The works in [5], [6] studied the backup server assignment problem with considering both failure probabilities of middleboxes and backup servers.

Since the dedicated protection requires a large amount of backup resources, similar to our work, the works in [7]–[10] considered the shared protection to reduce the required backup capacity. However, these works did not explicitly consider the unavailability of middlebox in their models. For example, the work in [7] considered maximizing the probability that the recovery for any set of failed middleboxes succeeds.

Recently, some works addressed availability-aware backup resource allocation models for service function chains (SFCs) [12], [13]. The work in [12] considered the software characteristics of virtual functions, where the backup sharing for basic and duty resources is clarified. The work in [13] studied the SFC mapping problem with availability requirements to minimize the required number of off-site backups. In this paper, we focus on the backup resource allocation for middleboxes instead of SFCs. More importantly, our work introduces a different way to estimate the middlebox availability. In [12], [13], the availability of a protected middlebox is calculated with considering the failure probabilities of related middleboxes and backup servers. This approach does not include some factors, such as the repair for a failed middlebox or backup server and the action after the shared protection fails, which also affect the availability. In this paper, the availability of each middlebox is estimated based on the queueing theory with comprehensively considering the failure, repair, and recovery behaviors of functions and backup servers.

III. MODEL AND PROBLEM DEFINITION

A. Unavailability model

Let F and S denote a set of functions and a set of backup servers, respectively. We consider that an element in

$W = F \cup S$ is either in an active state or failed state. We assume that each element, function or backup server, in W experiences failure events independently to other elements, where the failure events occur at an element based on a Poisson process when the element is active. For each element, the average failure rate of corresponding Poisson process can be estimated by sampling the status of element within a certain period. In literature, some studies introduced that the average failure rate of an element depends on its age or running time, which can be a key feature to classify elements in W to several classes, in each of which the elements have the same average failure rate [14], [15]. Let $H = [1, |H|]$ denote the set of classes for the elements in W ; an element in class $h \in H$ has the average failure rate of λ_h per unit time.

Once a failure event occurs to an element in W , it goes to the failed state and a procedure is started to repair the element itself to be active. For example, if a function or backup server fails due to a broken hardware component on its hosting physical machine, some methods, such as replacing the broken component, are taken to repair the physical machine, which makes the failed function or backup server return to active. We consider that the time to complete the repair procedure of a failed element in class $h \in H$ follows an exponential distribution with the average value of μ_h^{-1} unit time.

In literature, such as the works in [13], [16], the unavailability of function $i \in F$ is often considered as the ratio of its average unavailable time to the sum of its average available time and average unavailable time, which can be defined as,

$$Q_i = \frac{T_i}{T_i + T'_i}, \quad (1)$$

where T_i and T'_i represent the average unavailable time and the average available time of function i , respectively. Clearly, if there is no backup that is used to recover function i with class $h \in H$, its unavailability is $Q_i = \frac{\mu_h^{-1}}{\mu_h^{-1} + \lambda_h^{-1}}$.

B. Shared protection for functions

To protect a function against failure events, a backup server is assigned to protect it. We assume that, for different functions, the amounts of resources utilized for information synchronization or recovery on the backup servers are the same [5]–[8]. Backup server $j \in S$ can be assigned to protect at most c_j functions whose information required for recovery is updated to backup server j , where at most r_j functions can be recovered at the same time and $r_j \leq c_j$. Let x_{ij} denote a binary variable; if function $i \in F$ is protected by backup server $j \in S$, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

Figure 1 shows an example of backup allocation for three functions protected by two backup servers, where a link exists between function $i \in F$ and backup server $j \in S$ if $x_{ij} = 1$. Backup server j_1 can protect two functions and only one of them can be recovered at the same time. It indicates that functions i_1 and i_2 share the backup resource on backup server j_1 ; if functions i_1 and i_2 fail simultaneously, only one of them can be recovered at the same time.

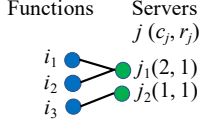


Fig. 1. Example of backup allocation with shared protection.

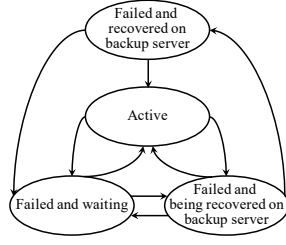


Fig. 2. State transition for each function.

C. Recovery on backup server

When a function in F is in the failed state, it needs to be recovered on a backup server before the completion of its repair procedure¹. Once the repair procedure is completed, the failed function becomes active and the corresponding recovered one if any on the backup server is released. In other words, the function recovered on a backup server is switched back to a function that becomes active. When a backup server in S is active, it can recover the failed functions if the recoveries do not exceed its capacity. When a backup server in S is in the failed state, it cannot recover any function.

For a function, there are two cases where the protection prepared for it fails. One is that a function and its backup server are in the failed states at the same time. For example, when both function i_3 and backup server j_2 in Fig. 1 are in the failed states, the prepared protection for function i_3 fails. The other is that a function is in the failed state and its preassigned backup server is active, but no capacity remains on the backup server to recover the failed function. For example, when function i_2 in Fig. 1 fails, if backup server j_1 is hosting failed function i_1 , the prepared protection for function i_2 fails.

To recover a failed function when its protection fails, different policies can be developed based on different application scenarios. In this paper, we consider a preassigned backup only policy (PBOP), where the failed function can only be recovered by its preassigned backup server. If the prepared protection fails, the failed function needs to wait. PBOP may be adopted in some scenarios with limited network bandwidth resources to avoid the additional transmissions for information required to recover failed functions [17]. An extension with considering different policies is left as a future work.

D. State transition and unavailable time of each function

Figure 2 shows the state transition for each function. There are three substates for a function in the failed state. When an active function fails, if its protection does not fail, the recovery procedure on the preassigned backup server starts; otherwise, the function waits to start the recovery on its backup server. When the preassigned backup server is active and has remaining capacity for recovery, the recovery procedure of failed function starts. In the case that multiple failed functions protected by the same backup server are in the waiting procedure, we consider randomly choosing some of them to

¹In this paper, for a failed function, the terminologies of *recover* and *repair* refer to recovering it on a backup server and making itself return to active through the repair procedure, respectively.

start the recoveries which do not exceed the maximum capacity of backup server. After the completion of recovery procedure, the failed function is recovered on the backup server and becomes available. For a failed function that is recovered or is being recovered on a backup server, if the backup server fails, the function goes to the waiting procedure. When the repair procedure of a failed function is completed, the failed function becomes active.

The operations to recover a failed function, such as launching a virtual machine and recreating the missing states, on an active backup server take time. We assume that the recovery time for a failed function on backup server $j \in S$ follows an exponential distribution with the average value of δ_j^{-1} unit time. The function is considered as unavailable during the waiting and recovery procedures. The time switching the function recovered on a backup server to a function that becomes active is assumed much shorter than the recovery time. In other words, the unavailable time of a function consists of two types of time, which are waiting time and recovery time.

E. Problem definition

The unavailability of a function depends on the backup allocation for all functions. Let $J = \max_{i \in F} Q_i$ denote the maximum unavailability among functions. The unavailability-aware shared backup allocation (UASBA) problem is defined as follows:

Problem 1: Given sets of functions and backup servers, average failure rate and average repair time for each function and backup server, and average recovery time on a backup server, find the optimal backup allocation for all functions to minimize the maximum unavailability among functions, or J .

To address UASBA, we first need to know how the unavailability of a function is for a given solution of backup allocation. We answer this question in Section IV by developing an analytical approach based on the queueing theory.

IV. ANALYSES FOR UNAVAILABILITY

Let $L_j = \{i \in F : x_{ij} = 1\}$ denote the set of functions protected by backup server $j \in S$; $L_j^h \subseteq L_j$ represents the set of all functions with class $h \in H$ in L_j . We consider backup server $j \in S$ and L_j as a group. Since a failed function can only be recovered by its preassigned backup server in PBOP, we analyze the unavailabilities of functions in a group independently of other groups. Two functions with the same classes in the same group have the same failure, repair, and recovery behaviors, which leads to the same unavailability.

We consider a state of functions of class $h \in H$ in the group containing backup server $j \in S$ as (m_h, n_h, o_h, p_h) , where m_h , n_h , o_h , and p_h represent the numbers of functions of class h which are active, waiting, being recovered, and recovered, respectively. Clearly, the values of m_h , n_h , o_h , and p_h satisfy $\sum_{h \in H} (m_h + n_h + o_h + p_h) = |L_j| \leq c_j$ and $\sum_{h \in H} (o_h + p_h) \leq r_j$. A state of backup server j is considered as (q) , where $q \in \{0, 1\}$; $q = 1$ and 0 mean that backup server j is in the active and failed states, respectively. A state of the whole group consists of the substates of $|H|$

classes of functions and backup server, which is expressed by $\Gamma = (m_1, n_1, o_1, p_1, \dots, m_{|H|}, n_{|H|}, o_{|H|}, p_{|H|}, q) \in U_j$, where U_j denotes a set of all feasible states for the group containing backup server $j \in S$.

We analyze the case of $|H| = 1$ to present the basic idea of estimating the unavailability of a function based on the queueing theory. The cases of $|H| > 1$ can be addressed based on the result of $|H| = 1$. In the case of $|H| = 1$, each element in W has the same average failure rate of λ and the same average repair time of μ^{-1} . A state of the group containing backup server $j \in S$ is expressed by (m, n, o, p, q) .

A. Number of feasible states

We analyze the number of feasible states of group containing backup server $j \in S$ in two conditions. One is $|L_j| \geq r_j$, and the other is $|L_j| < r_j$.

When $|L_j| \geq r_j$, we consider two cases; $q = 1$ in case 1 and $q = 0$ in case 2. For case 1, there are two sub cases. In case 1a, $o + p \in [0, r_j - 1]$. Since backup server j has remaining capacity to recover one or more failed functions, $n = 0$ in case 1a. When $o + p = t \in [0, r_j - 1]$, $m = |L_j| - t$ and there are $t + 1$ possibilities for the values of o and p . Hence, in case 1a, there are $\frac{r_j^2 + r_j}{2}$ feasible states. In case 1b, $o + p = r_j$ and $m + n = |L_j| - r_j$. There are $r_j + 1$ and $|L_j| - r_j + 1$ possibilities for the values of o and p and for the values of m and n , respectively. Hence, in case 1b, there are $(r_j + 1)(|L_j| - r_j + 1)$ feasible states. Consequently, $\frac{r_j^2 + r_j}{2} + (r_j + 1)(|L_j| - r_j + 1)$ feasible states exist in case 1. For case 2, $o + p = 0$ and $m + n = |L_j|$, which leads to $|L_j| + 1$ feasible states. Therefore, in total $\frac{r_j^2 + r_j}{2} + (r_j + 1)(|L_j| - r_j + 1) + |L_j| + 1$ feasible states exist when $|L_j| \geq r_j$.

When $|L_j| < r_j$, the backup server can recover all functions in L_j at the same time if it is not in the failed state. For the case of $q = 1$, $m + o + p = |L_j|$ and $n = 0$, which indicates $\frac{|L_j|^2 + 3|L_j| + 2}{2}$ feasible states. For the case of $q = 0$, $m + n = |L_j|$ and $o + p = 0$, which leads to $|L_j| + 1$ feasible states. Therefore, in total $\frac{|L_j|^2 + 5|L_j| + 4}{2}$ feasible states exist when $|L_j| < r_j$.

B. State transition for (m, n, o, p, q)

Figure 3 shows the state transition for system state $(m, n, o, p, q) \in U_j$, $j \in S$, where nine types of states can be incoming to state (m, n, o, p, q) , which can be outgoing to nine types of states. Table I describes the condition and transfer rate for each type of transition, where $l = \min(n, r_j)$. We number the types of states from 1 to 18. For type 8 of $(m, n - o' - p', o', p', q + 1)$, there are $l + 1$ states incoming to state (m, n, o, p, q) with considering the conditions of $o' + p' = l$. Except for type 8, each type corresponds to one state incoming to or outgoing from state (m, n, o, p, q) .

Note that each type of transition corresponds to one or several situations, each of which contains one or several changes on the states of functions and backup server. Both transitions of types 1 and 10 correspond to two situations. In one situation, a waiting function becomes active. In the other situation,

a function in the recovery procedure becomes active and a waiting function starts the recovery procedure. Each of other types corresponds to one situation. For example, the transition of type 2 indicates that a function in the recovery procedure becomes active and there is no waiting function.

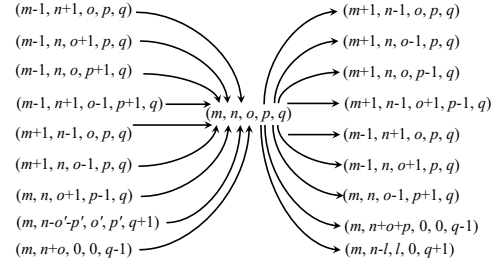


Fig. 3. State transition for system state (m, n, o, p, q) .

C. Equilibrium States

Let $P(m, n, o, p, q)$ represent the probability that the system is in state $(m, n, o, p, q) \in U_j$, $j \in S$. In the equilibrium state, the total incoming flows to state (m, n, o, p, q) are equivalent to the total outgoing flows from state (m, n, o, p, q) . We obtain an equilibrium equation for each state. The total probability of all states equals one, which is given by, $\sum_{(m, n, o, p, q) \in U_j} P(m, n, o, p, q) = 1$. By solving the multiple equations, we obtain the probability for each state.

D. Unavailability of function

For state $(m, n, o, p, q) \in U_j$, $j \in S$, there are $n + o$ unavailable functions. The average number of unavailable function in the group is computed by $\sum_{(m, n, o, p, q) \in U_j} (n + o)P(m, n, o, p, q)$. For types 14 and 15, the transition outgoing from state (m, n, o, p, q) to its corresponding state indicates that a function becomes unavailable with the rate of $m\lambda$. For type 17, the transition indicates that a backup server fails and p functions become unavailable with the rate of λ . Hence, per unite time, $\sum_{(m, n, o, p, q) \in U_j} (m\lambda + pq\mu)P(m, n, o, p, q)$ functions become unavailable. By using Little's formula [18], the average unavailable time of function $i \in L_j$ is expressed by, $T_i = \frac{\sum_{(m, n, o, p, q) \in U_j} (n + o)P(m, n, o, p, q)}{\sum_{(m, n, o, p, q) \in U_j} (m\lambda + pq\mu)P(m, n, o, p, q)}$. Similarly, the average available time of function i in the group is expressed by, $T'_i = \frac{\sum_{(m, n, o, p, q) \in U_j} (m + p)P(m, n, o, p, q)}{\sum_{(m, n, o, p, q) \in U_j} [(n + o + p)\mu + o\delta_j]P(m, n, o, p, q)}$. Based on T_i and T'_i , we obtain the unavailability of function i by using (1).

V. HEURISTIC ALGORITHM

We introduce a heuristic algorithm based on SA [19] to solve the UASBA problem. Let D^{init} , D^{term} , and ρ denote three given parameters in the SA algorithm, where $D^{\text{init}} > D^{\text{term}} > 0$ and $0 < \rho < 1$; D represents a running variable to indicate the "temperature" in SA. At the beginning of SA algorithm, $D = D^{\text{init}}$ and a feasible solution consisting of $x_{ij}, i \in F, j \in S$ is randomly set; the maximum unavailability among functions in each feasible solution is computed by the approach provided in Section IV. Then, in each iteration,

TABLE I
STATES TRANSITION INCOMING TO AND OUTGOING FROM STATE (m, n, o, p, q) .

Direction	Type, k	State	Transfer rate	Condition
Incoming states	1	$(m-1, n+1, o, p, q)$	$(n+1+o)\mu$	$m \geq 1$ and $o+p=r_j$ $m \geq 1$ and $q=0$
	2	$(m-1, n, o+1, p, q)$	$(o+1)\mu$	$m \geq 1, o+1+p \leq r_j$, and $q=1$
	3	$(m-1, n, o, p+1, q)$	$(p+1)\mu$	$m \geq 1, o+p+1 \leq r_j$, and $q=1$
	4	$(m-1, n+1, o-1, p+1, q)$	$(p+1)\mu$	$m \geq 1, o \geq 1$, and $o+p=r_j$
	5	$(m+1, n-1, o, p, q)$	$(m+1)\lambda$	$n \geq 1$
	6	$(m+1, n, o-1, p, q)$	$(m+1)\lambda$	$n=0$ and $o \geq 1$
	7	$(m, n, o+1, p-1, q)$	$(o+1)\delta_j$	$p \geq 1$
	8	$(m, n-o'-p', o', p', q+1)$	λ	$q=0$
	9	$(m, n+o, 0, 0, q-1)$	μ	$p=0$ and $q=1$
Outgoing states	10	$(m+1, n-1, o, p, q)$	$(n+o)\mu$	$n \geq 1$
	11	$(m+1, n, o-1, p, q)$	$o\mu$	$n=0$ and $o \geq 1$
	12	$(m+1, n, o, p-1, q)$	$p\mu$	$n=0$ and $p \geq 1$
	13	$(m+1, n-1, o+1, p-1, q)$	$p\mu$	$n \geq 1$ and $p \geq 1$
	14	$(m-1, n+1, o, p, q)$	$m\lambda$	$m \geq 1$ and $o+p=r_j$ $m \geq 1$ and $q=0$
	15	$(m-1, n, o+1, p, q)$	$m\lambda$	$m \geq 1, o+1+p \leq r_j$, and $q=1$
	16	$(m, n, o-1, p+1, q)$	$o\delta_j$	$o \geq 1$
	17	$(m, n+o+p, 0, 0, q-1)$	λ	$q=1$
	18	$(m, n-l, l, 0, q+1)$	μ	$q=0$

$D = \rho D$ and a new feasible solution is generated by applying a random change to the existing solution. If the objective value of new solution, J' , is less than that of the existing one, J , the new solution is accepted and replaces the existing one; otherwise, the new solution is accepted with a probability of $e^{(\frac{J-J'}{D})}$. As a result, the SA algorithm can avoid a local minimum solution. The algorithm is terminated when $D \leq D^{\text{term}}$.

We analyze the computational time complexity of SA. It takes $O(|F||S|)$ to generate a feasible solution. Given a feasible solution, as introduced in Section IV, we create and solve the multiple linear equations for each group to obtain the maximum unavailability. Consider the maximum number of feasible states, or the maximum number of variables in multiple equations, among all groups as $O(A)$; $A = r^2$ in the case of $|H| = 1$, where $r = \max_{j \in S} r_j$. The computational time complexity of creating the multiple equations for a group is $O(A)$. Solving the multiple equations for a group takes $O(A^2)$ when the Gauss-Seidel algorithm [20] is adopted. Therefore, computing the maximum unavailability for a given solution takes $O(A^2|S|)$, which is the computational time complexity for each iteration of the SA algorithm.

For the approximation performance of SA algorithm, there is a tradeoff between the number of iterations and the accuracy of solution. A more accurate solution can be obtained by increasing the number of iterations, or increasing the value of ρ .

VI. NUMERICAL RESULTS

We first introduce a conventional model as a baseline model. Then we compare the proposed model with the baseline model in terms of the maximum unavailability among functions. The performance dependencies on the backup server capacity, failure rate, average repair time, and average recovery time are evaluated. We use Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory through our evaluations.

A. Baseline model

Without comprehensively considering the failure, repair, and recovery behaviors of functions and backup servers, the conventional backup allocation models with shared protection are

not explicitly aware of unavailability. Instead of considering the function unavailability directly, the work in [7] presented a survivability-aware backup allocation model, which is considered as the baseline model.

In the baseline model, element $k \in W$ with class $h \in H$ is associated with a failure probability, which is considered as $\alpha_k = \frac{\mu_h^{-1}}{\mu_h^{-1} + \lambda_h^{-1}}$. The survivability for a backup allocation is defined as the probability that the protection for a set of failed functions succeeds. The baseline model aims to find the backup allocation that maximizes the survivability. We solve the baseline model by modifying the SA algorithm introduced in Section V. After the results of backup allocation are obtained, the exact maximum unavailability among functions is computed by the approach provided in Section IV.

B. Experiment Setup

A network with 100 middleboxes, or $|F| = 100$, is considered in our experiments. The number of backup servers is considered as $|S| = 20$. Each result is an average over 500 trials, in each of which we randomly select two integers from the range of $[1, M]$ as the values of c_j and r_j , $j \in S$, respectively, with considering the condition of $c_j \geq r_j$, where M denotes the upper bound of range. Similarly, the average recovery time of δ_j^{-1} on backup server $j \in S$ is randomly set within a range of $[L, N]$ in each trial, where L and N denote the lower and upper bounds of range, respectively. We set $D^{\text{init}} = 10^7$, $D^{\text{term}} = 10^{-5}$, and $\rho = 0.99$ for the SA algorithm.

According to the studies in [2], the median and 95th percentile of time between two adjacent failures for devices in a network are 10^4 and 10^6 [s], respectively; the median and 95th percentile of repair time are 10^3 and 10^5 [s], respectively. In the experiments in [11], the recovery time on a backup server varies from 30 to 10^2 [s]. In our evaluations, the parameters are set with refereeing the above investigations.

C. Evaluation

Figure 4 shows the maximum unavailabilities obtained by the proposed and baseline models for different values of

upper bound of capacity range of backup servers, where $\lambda = 10^{-4}$, $\mu^{-1} = 10^3$, and the range of $\delta_j^{-1}, j \in S$, is considered as $[30, 10^2]$. We observe that, as the value of M increases, the maximum unavailability obtained by the proposed model decreases. This is because, with a larger value of $r_j, j \in S$, a protected function can have less probability to be in the waiting procedure. Compared to the baseline model, the proposed unavailability-aware model reduces the maximum unavailability 9% in average.

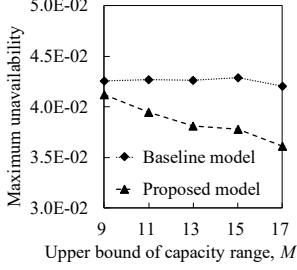


Fig. 4. Comparison among maximum unavailabilities obtained by different models for different values of M .

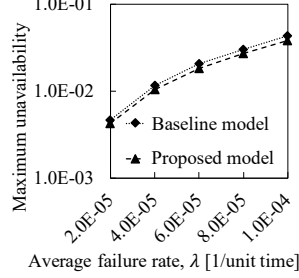


Fig. 5. Dependency of maximum unavailabilities obtained by different models on average failure rate.

Table II shows the computation times of proposed and baseline models to obtain the results shown in Fig. 4. The computation time of proposed model increases as the value of M increases, which fits our theoretical analyses in Section V. The results observe that the computation time of proposed model is 13 times longer in average than that of baseline model. However, since it is much smaller than the investigated time between two adjacent failures, repair time, and recovery time, the proposed model can be acceptable in practical applications in terms of the computation time.

TABLE II
COMPUTATION TIME [S] OF DIFFERENT MODELS.

M	Proposed model	Baseline model
9	1.288	0.133
11	1.531	0.129
13	1.834	0.135
15	2.098	0.139
17	2.390	0.143

Figure 5 shows the dependency of maximum unavailabilities obtained by the proposed and baseline models on the average failure rate, where $M = 15$, $\mu^{-1} = 10^3$, and the range of $\delta_j^{-1}, j \in S$, is considered as $[30, 10^2]$. As the average failure rate increases, the maximum unavailabilities of both models increase. The proposed model reduces the maximum unavailability 11% in average compared to the baseline model.

Figure 6 compares the maximum unavailabilities obtained by the proposed and baseline models for different values of average repair time, where $M = 15$, $\lambda = 10^{-4}$, and the range of $\delta_j^{-1}, j \in S$, is considered as $[30, 10^2]$. We observe that the maximum unavailabilities of both models increase as the average repair time increases. On one hand, a larger average repair time for a backup server leads to a higher probability that a failed function is in the waiting procedure. On the

other hand, the time that a failed function occupies a backup server becomes long as the average repair time of a function increases. As a result, the probability that a failed function is in the waiting procedure becomes high. The proposed model reduces the maximum unavailability 10% in average compared to the baseline model.

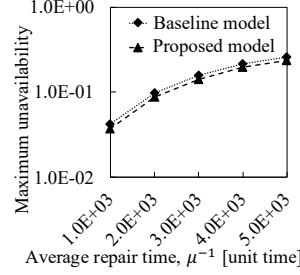


Fig. 6. Comparison among maximum unavailabilities obtained by different models for different values of average repair times.

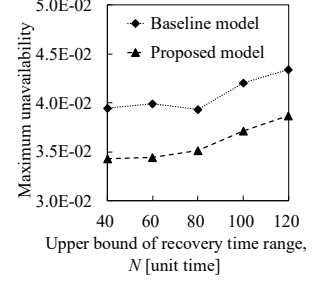


Fig. 7. Comparison among maximum unavailabilities obtained by different models for different values of N .

Figure 7 shows the comparison among maximum unavailabilities obtained by the proposed and baseline models for different values of upper bound of repair time range, where the lower bound is set to $L = 30$, $M = 15$, $\lambda = 10^{-4}$, and $\mu^{-1} = 10^3$. We observe that, as the value of N increases, it takes longer time to recover a failed function, which increases the maximum unavailabilities for both models. In this evaluation, compared to the baseline model, the proposed model reduces the maximum unavailability 12% in average.

VII. CONCLUSION

This paper proposed an unavailability-aware backup allocation model with the shared protection for middleboxes with comprehensively considering the failure, repair, and recovery behaviors of functions and backup servers. The backup resources on a backup server can be shared by multiple functions. The proposed model aims to find the backup allocation for all functions to minimize the maximum unavailability among functions. The multiple situations of failure, repair, and recovery of functions and backup servers lead to several different states for each function. The unavailability of function was estimated through analyzing all states that a function can be in. We developed an analytical approach based on the queueing theory to compute the unavailability of middlebox for a given backup allocation. Based on the analytical approach, the SA heuristic was introduced to solve the backup allocation problem. The performance dependencies on the backup server capacity, failure rate, average repair time, and average recovery time were evaluated. We compared the proposed model with the baseline model. The results observed that, compared to the baseline model, the proposed unavailability-aware model reduces the maximum unavailability 11% in average in our examined scenarios.

ACKNOWLEDGMENT

This work was supported in part by JSPS KAKENHI, Japan, under Grant Number 18H03230.

REFERENCES

- [1] R. Mijumbi *et al.*, “Network function virtualization: state-of-the-art and research challenges,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236-262, 2016.
- [2] R. Potharaju *et al.*, “Demystifying the dark side of the middle: a field study of middlebox failures in datacenters”, in *Proc. IMC*, 2013, pp. 9-22.
- [3] J. Sherry *et al.*, “Rollback-recovery for middleboxes”, in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 227-240, 2015.
- [4] J. Fan *et al.*, “A framework for provisioning availability of NFV in data center networks,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246-2259, 2018.
- [5] F. He *et al.*, “Optimization model for backup resource allocation in middleboxes,” in *Proc. of IEEE CloudNet*, 2018.
- [6] F. He *et al.*, “Optimization model for backup resource allocation in middleboxes with importance,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1742-1755, 2019.
- [7] Y. Kanizo *et al.*, “Optimizing virtual backup allocation for middleboxes,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2759-2772, 2017.
- [8] Y. Kanizo *et al.*, “Designing optimal middlebox recovery schemes with performance guarantees,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2373-2383, 2018.
- [9] F. He *et al.*, “Robust optimization model for backup resource allocation in cloud provider,” in *Proc. IEEE ICC*, 2018.
- [10] F. He *et al.*, “Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures,” in *Proc. DRCN*, 2019.
- [11] T. Sato *et al.*, “Implementation and testing of failure recovery based on backup resource sharing model for distributed cloud computing system,” in *Proc. IEEE CloudNet*, 2018.
- [12] D. Li *et al.*, “Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy,” *IEEE Trans. Netw. Service Manag.*, 2019.
- [13] J. Fan *et al.*, “Availability-aware mapping of service function chains,” in *Proc. IEEE INFOCOM*, 2017, pp. 1-9.
- [14] T. Kuwabara *et al.*, “Calculation method of failure probabilities of optical fiber,” *Journal of lightwave technology*, vol. 11, no. 7, pp. 1132-1138, 1993.
- [15] S. Sekigawa *et al.*, “Expected Capacity Guaranteed Routing based on Dynamic Link Failure Prediction,” in *Proc. of ICNC*, 2019, pp. 170-174.
- [16] J. Fan *et al.*, “Carrier-grade availability-aware mapping of service function chains with on-site backups,” in *Proc. IEEE IWQoS*, 2017, pp. 1-10.
- [17] B. Yang *et al.*, “Algorithms for fault-tolerant placement of stateful virtualized network functions,” in *Proc. IEEE ICC*, 2018.
- [18] J.D.C. Little, “A proof for the queuing formula: $L = \lambda W$,” *Oper. Res.*, vol. 9, no. 3, pp. 383-387, 1961.
- [19] T. Segaran, *Programming collective intelligence: building smart web 2.0 applications*, O'Reilly Media, Inc, 2007.
- [20] A. Björck, *Numerical methods in matrix computations*. Cham, Switzerland: Springer, 2015.