Received 7 February 2025; revised 24 March 2025; accepted 15 April 2025. Date of publication 18 April 2025; date of current version 1 May 2025.

Digital Object Identifier 10.1109/OJCOMS.2025.3562234

Analysis of Unavailability in Middleboxes With Multiple Backup Servers Under Shared Protection

NAOHIDE WAKUDA (Graduate Student Member, IEEE), RYUTA SHIRAKI (Member, IEEE), AND EIJI OKI (Fellow, IEEE)

Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

CORRESPONDING AUTHOR: E. OKI (e-mail: oki@i.kyoto-u.ac.jp)

A part of this paper was presented in [1] at the 8th IEEE International Conference on the Design of Reliable Communication Networks, May 2024 [DOI: 10.1109/DRCN60692.2024.10539160].

ABSTRACT Middlebox functions, implemented as software on general-purpose servers via network function virtualization, require reliable protection mechanisms to ensure service continuity. Assessing the unavailability of these functions is critical, as failures can lead to significant service disruptions. However, existing analytical models primarily assume that a function is protected by at most one or two backup servers, limiting their applicability in scenarios requiring higher resilience. To address this limitation, this paper proposes an analytical model for evaluating the unavailability of middlebox functions under a multiple-backup shared protection strategy, where multiple backup servers protect one or more functions. Our model allows each function to be protected by multiple backup servers, ensuring availability while ensuring that each backup server can simultaneously recover at most one function. Utilizing a Markov chain, we analyze state transitions and establish equilibrium-state equations, providing an analytical foundation for evaluating the performance of the multiple-backup shared protection strategy. Numerical results demonstrate that this strategy significantly enhances availability, reducing unavailability by up to 72.3% compared to the single-backup shared protection strategy in the scenarios examined. Our study provides a detailed analysis of backup allocation strategies, focusing on their impact on function availability and offering more profound insights into their effectiveness through theoretical properties and performance comparisons with existing strategies. Our evaluation reveals that the multiple-backup shared protection strategy reduces unavailability by up to 64.8% compared to the single-backup shared protection strategy in the examined allocation cases.

INDEX TERMS Analytical model, middleboxes, unavailability, shared protection, Markov chain.

I. INTRODUCTION

MIDDLEBOXES provide a range of network functions, such as firewalls, network address translators, intrusion detection systems, and load balancers, which are critical to the network infrastructure. Traditionally, a middlebox operates on specialized hardware designed for specific network functions. By utilizing network function virtualization (NFV) technology [2], middleboxes can operate as software on general-purpose servers. Hence, a middlebox function works as a service function instance (SFI) using NFV technology. Henceforth, a function used in this paper refers to an SFI. This approach allows for optimized utilization of hardware resources within the network. Furthermore, NFV technology supports the dynamic deployment of middleboxes, thereby increasing the flexibility of network management and operations.

Middleboxes can experience failures for various reasons, including hardware faults, configuration errors, and connectivity issues. When a middlebox becomes unavailable due to these incidents, it significantly degrades the quality of network services that rely on it [3]. Consequently, the absence of network functions constitutes a crucial metric connected to the overall quality of network services.

One approach to mitigate network function unavailability is deploying backup servers capable of restoring functionality in case of an unrecovered failure. This approach employs two protection strategies: dedicated protection and shared protection. The research in [4], [5], [6] addressed

allocating a dedicated backup server for each network function. The dedicated protection strategy designates a distinct backup server for each function, ensuring recovery if the backup server remains operational. Consequently, the dedicated protection strategy exhibits a higher probability of function recovery than the shared protection approach, thereby preserving the network service quality. However, it is worth noting that the dedicated protection strategy demands a substantial amount of computational resources, as the backup server cannot concurrently recover other functions while protecting an active one. Conversely, in the shared protection strategy, each backup server can protect one or more functions. Still, the guarantee of dedicated protection for each function by the backup server is limited due to the finite capacity of the backup server.

Several studies have addressed shared protection [7], [8], [9]. The work in [7] investigated resource allocation under shared protection aware of function survivability. The work analyzed the resource allocation problems using graph theory and introduced algorithms that maximize the oriented metrics of function survivability. Studies [8], [9] investigated the performance of shared protection, analytically computing exact function unavailability. In particular, concerning shared protection strategies aimed at minimizing function unavailability, the research in [8] introduced a model for optimizing the allocation to minimize the maximum unavailability among middlebox functions. The work delineated an analytical method for computing exact function unavailability employing Markov chain analysis of state transitions. It specifically focused on scenarios where a single backup server protects each function. Furthermore, the research in [9] presented an analytical model in which backup servers are capable of protecting two functions and can recover one of them. The work in [10], [11] addressed shared protection in the case of multiple backup servers and addressed a model aimed at minimizing the required backup capacity. However, the work does not provide a model for calculating unavailability. No work deals with an analytical model to compute function unavailability where more than two backup servers can protect a function.

Two key questions arise: 1) How can we calculate the unavailability of functions when multiple backup servers participate in shared protection? 2) How much does this shared protection with multiple backup servers reduce unavailability compared to the existing one in which one backup server protects one or more functions? This paper addresses these questions.

This paper proposes an analytical model to compute the unavailability of middlebox functions in which one or more backup servers protect one or more functions and one backup server can recover at most one function, which we call a multiple-backup shared protection strategy. On the other hand, in this paper, the single-backup shared protection strategy represents the shared protection in which one backup server protects one or more functions, and it can recover multiple functions simultaneously. The proposed model analytically computes the unavailability of functions by thoroughly examining state transitions involving both functions and backup servers for the multiple-backup shared protection. When an active function experiences a failure, an active and idle backup server can recover the failed function. During the recovery process by the backup server, the function remains in a failed state. Upon completing the function's repair, it regains its active status. The model calculates the unavailability of functions by thoroughly analyzing the state transitions of a group in the multiple-backup shared protection strategy. We conduct performance evaluations using the proposed model. Numerical results show that the multiple-backup shared protection strategy reduces the unavailability by 7.80–72.3% compared to the single-backup shared protection strategy in our examined cases. The single-backup shared protection strategy was previously examined by one of the authors of this paper in earlier studies [8], [12], [13]. In contrast to those works, which concentrated on the single-backup approach, our research offers a more comprehensive analysis of backup allocation strategies. Specifically, we explore their influence on function availability and provide deeper insights into their effectiveness by deriving theoretical properties and comparing performance with existing methods. Numerical results demonstrate that the multiple-backup shared protection strategy minimizes unavailability by a maximum of 64.8% compared to the single-backup shared protection strategy in our examined allocation scenarios.

This paper is an expanded edition of [1] with various additions described as follows. We thoroughly review existing research on shared protection, middleboxes, and Markov chains by comparing our work with existing studies. We provide a thorough explanation of the proposed model, detailing the states of both functions and servers. We explain the state transitions by categorizing them into different types, providing an example of a state-transition diagram to support the explanation. We compare the unavailability obtained by the analytical model and simulation. We investigate backup allocation regarding unavailability, exploring the nature of backup allocation, where we derive three properties by providing their proof. We compare the backup-allocation performances between the multiple-backup and the single-backup strategies.

The remainder of this paper is structured subsequently. Section II outlines the related work in this field. Section III explains the proposed model, covering shared protection for functions, unavailability, and state transitions. Section IV analyzes the model, addressing the count of feasible states, state transition of a group, and equilibrium-state equations. Section V shows the numerical results. Section VI investigates backup allocation regarding unavailability. Section VII wraps up the paper.

II. RELATED WORK

This section describes related works, grouped into three areas: shared protection, middleboxes, and Markov Chains.

Similar to our work, the works in [7], [8], [9], [12], [13], [14], [15], [16], [17], [18], [19], [20] addressed shared protection. As we mentioned for the work in [7], [8], [9] in Section I, the distinction between these studies and our research is that while our work develops an analytical model for two or more backup servers, these previous studies handled a model for only one or, in specific configurations, multiple backup servers. The work in [12], [13] introduced a model for shared protection with multiple backup servers and analyzed the model using a Markov chain. While the work introduced a model for cases with two backup servers without a constraint on the number of functions that a backup server can protect, our research presents a model adaptable to cases with two or more backup servers. The work in [14] introduced a backup allocation model considering both virtual and physical machine failures to minimize the maximum unavailability of functions. The work separates functional failures between virtual and physical machines. It employs a single-backup shared protection strategy, which differs from our work. The work in [15] investigated the design of a protection strategy aimed at minimizing the needed backups through resource sharing. The work in [16] integrated various types of resources necessary for the backup allocation model in network functions. The work in [15], [16] are different in that our work addresses the unavailability of middlebox functions, whereas these works address the design of protection strategy and backup resource integration. The work in [17], [18], [19], [20] addressed the design of the protection strategy of service function chains. They are different in that our work addresses the unavailability of middlebox functions. In contrast, they addressed the design of the protection strategy of service function chains but did not use Markov chains to analyze the model.

Our study is related to the development of a framework for fault-tolerant middleboxes [6], [21], [22]. The work in [6] considered the importance of middlebox functions and sought to minimize the worst-weighted unavailability by assigning backup servers to functions. The work does not consider the recovery and repair times of middlebox functions and backup servers in assessing the unavailability. The work in [21] introduced a system-level framework that uses the flow-centric structure to improve middlebox availability with low overhead. In [22], a framework with rollback recovery for middleboxes was presented, reducing protection overhead by using ordered logging and parallel release. These works in [21], [22] are different in that our work addresses computing the unavailability of middlebox functions, whereas these works focused on the development of a framework for middleboxes.

Similar to our study, several studies utilize Markov chains [23], [24], [25], [26]. The universal generating function (UGF) method is used to calculate service function chain (SFC) availability by using a continuous-time Markov chain [23], [24], [25], where the distribution of computing resources achieves reliable SFCs without considering shared protection. In [23], core networks are designed using the Internet protocol multimedia subsystem to achieve the required network slice availability through the UGF method. The work in [24] analyzed multitenant service function chain availability in NFV environments by introducing a multidimensional UGF method. The work in [25] introduced a virtual network function (VNF) placement model to minimize deployment costs while ensuring availability for service function chains, utilizing the multidimensional UGF method for computing SFC availability with multiple VNFs on a server. Our work also employs a continuous-time Markov chain. Whereas the works in [23], [24], [25] considered allowing the distribution of computing resources without adopting shared protection, our work addresses shared backup protection. The work in [26] introduced AnalyticalBP, a precise analytical continuous-time Markov chain model for blocking probabilities in spectrally-spatially elastic optical networks, addressing the challenge of crosstalk. Similar to our research, the work uses a Markov chain model to investigate situations where the next state depends only on the current one. Different from [18], looking at blocking probabilities, our study focuses on figuring out unavailability.

III. PROPOSED ANALYTICAL MODEL

A. SHARED PROTECTION FOR FUNCTION

Let $G \subseteq S$ be a group of servers that serve as backups for the same functions. We denote the set of functions backed up by G as $L_G \subseteq F$. Our analysis treats G and L_G as a connected component. Table 1 provides a comprehensive overview of the frequently utilized notations.

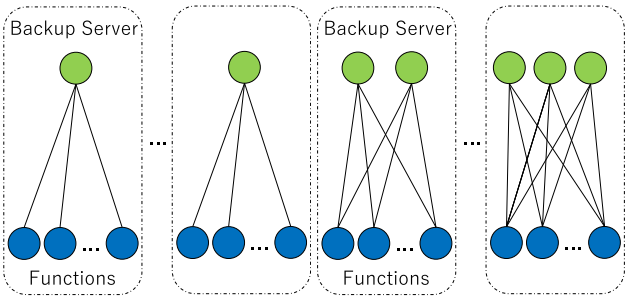
We assume the supply of ample and dependable networking resources. This research investigates how backup servers are assigned to different functions, taking into account the limited computing resources and the varied processes that may take place at network nodes, functions, or backup servers, which in turn affect the availability of functions.

Following the works of [27], [28], [29], we also assume that the time between the same type of events for each function or server follows an exponential distribution, which is a specific case of the Weibull and gamma distributions. Our analysis utilizes a Markov chain to model these events. Moreover, the study in [30] presented that an exponential distribution can effectively represent the time intervals between failures for most cloud services.

Fig. 1a illustrates an instance of the single-backup shared protection, which the analytical model presented in [8] handles. Fig. 1b shows an example of the multiple-backup shared protection, which the proposed analytical model handles. In the multiple backup shared protection, backup servers are designated to protect multiple functions. It is

TABLE 1. List of frequently used notations.

Notation	Meaning
F	Set of functions
S	Set of backup servers
G	Group of backup servers for the same functions
L_G	Set of functions backed up by group G
Γ	A system state of the group containing backup group G
U	Set of all feasible states in Γ
λ_{f}	Given parameter indicating average failure rate of function
μ_{f}	Given parameter indicating average recovery rate of function
λ_{s}	Given parameter indicating average failure rate of backup server
μ_{s}	Given parameter indicating average recovery rate of backup server
δ	Given parameter indicating average repair rate
m	Number of active functions in a group
n	Number of failed and waiting functions in a group
o	Number of functions which are being recovered by backup servers in a group
p	Number of functions which are recovered by backup servers in a group
q	Number of active backup servers in a group



(a) Single-backup shared protection.

(b) Multiple-backup shared protection.

FIGURE 1. Example of single-backup shared protection and multiple-backup shared protection.

crucial to note that each backup server can recover only one function at a time.

B. UNAVAILABILITY

$$Q_f = \frac{T_f^{\mathrm{u}}}{T_f^{\mathrm{a}} + T_f^{\mathrm{u}}}. \tag{1}$$

The unavailability of a function f in a group is the same and denoted as Q_f, where T_f^{a} and T_f^{u} represent the average available time of function f and the average unavailable time of function f, respectively. The unavailability of a function is a metric that is often used to evaluate the performance of a protection strategy. In this study, we compute the unavailability of functions in a group.

In this study, we also investigate minimizing the maximum unavailability among functions to obtain the best allocation of functions and backup servers with the minimum unavailability among all possible allocations. The maximum unavailability among functions is denoted as $Q_{\max}$, and is expressed by:

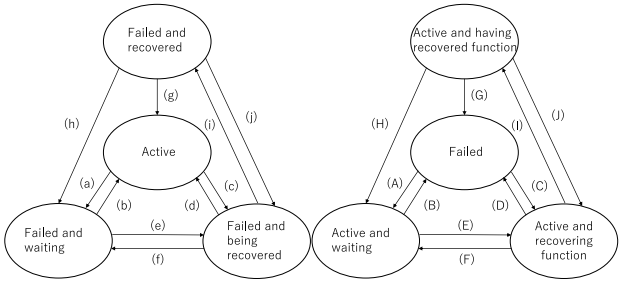
$$Q_{\max} = \max_{f \in F} Q_f. \tag{2}$$

The maximum unavailability among functions is a metric that is often used to evaluate the performance of a protection strategy [8], [14].

C. STATE TRANSITION

When an active function fails, it goes to the failed and waiting state if there is no available backup server or to the failed and being recovered state if there is at least one available backup server. Here, waiting means waiting for a backup server to become available. Once a backup server becomes available, a random selection is made from the functions in the failed and waiting state. Then the available backup server starts a recovery procedure for the selected function, and the function goes to the failed and being recovered state. This random approach offers ease of modeling and streamlines network administration and operations for service providers. Upon completion of the recovery procedure, the previously failed function is successfully recovered on the backup server, becomes available again, and goes to the failed and recovered state. If a backup server fails while a failed function is being recovered or has been recovered on it, the function goes to the failed and waiting state or to the failed and being recovered state, which depends on the backup server's state. Once the repairing procedure for a failed function is finished, the function goes to the active state. Throughout the waiting and recovery state, the function is considered unavailable. Notably, the time required to switch from a recovered function on a backup server to an active function is presumed to take considerably less time than the actual recovery process. In essence, the unavailable time of a function comprises two distinct components: the waiting time and the recovery time.

Fig. 2b shows the state-transition diagram of a server. When a failed server gets repaired, it goes to the active and waiting state or to the active and recovering function state, which depends on the number of functions in the failed and waiting state. Here, waiting means waiting for a function to fail. Once a function fails when backup servers are in the active and waiting state, a random selection is made from the servers, and the selected server starts recovering the function and goes to the active and recovering function state. After the recovery procedure is completed, the previously failed function is recovered on the backup server and the server goes to the active and having recovered function state. If a backup server fails, the backup server goes to the failed state. Once the repairing procedure for a failed function is finished, the backup server goes to the active and waiting state or to the active and recovering function state, which



(a) State-transition diagram of function.

(b) State-transition diagram of server.

FIGURE 2. State-transition diagram of function and server.

depends on the number of functions in the failed and waiting state.

Table 2 shows the state-transition cases for each function and server, respectively. A trigger event of a function or a server is the event that occurs due to a state change by itself, but not triggered by any other event. A trigger event can invoke an induced event.

Table 3 shows the states that a group can take by combining trigger events and induced events in Table 2. Considering the combination of trigger events and induced events, there are 13 possible state transitions that a group can take.

D. PROBLEM DEFINITION FOR BACKUP ALLOCATION

Backup allocation is a critical part that affects the unavailability of functions. The minimum unavailability backup allocation (MUBA) problem is stated as:

Problem: Given sets of functions and backup servers, with each function and backup server characterized by their average failure and repair rate, along with the average recovery time on a backup server, the objective is to determine the backup allocation that minimizes the maximum unavailability of function in the group denoted by $Q_{\max}$.

To address the MUBA problem, it is crucial to ascertain the function's unavailability for a given backup allocation solution.

IV. ANALYSIS FOR UNAVAILABILITY BASED ON QUEUEING THEORY

A. NUMBER OF FEASIBLE STATES

A system state of the group containing backup server group G is expressed by $\Gamma = (m, n, o, p, q)$, where m, n, o, and p are the number of functions that are active, waiting, being recovered, and recovered, respectively, and q is the number of active backup servers in G. Clearly, the values of m, n, o, p, and q satisfy $m + n + o + p = |L_G|$, $o + p \leq q$, and $q \leq |G|$.

We analyze the number of feasible states. First, we consider the number of feasible states in the condition of $|L_G| \geq q$. In considering the feasible states of Γ, when $o+p \in [0, q-1]$, then $n = 0$ since backup servers can start recovering at least another failed function. When $o + p = t$ and t falls within the interval $[0, q-1]$, the value of m is given by $m = |L_G| - t$. In this scenario, there are $t + 1$ distinct pairs (o, p). As a result, when $o+p \in [0, q-1]$, the number of feasible state is $\sum_{t=0}^{q-1}(t+1) = \frac{(q+1)q}{2}$. For the case where $o + p = q$, there exist $q + 1$ and $|L_G| - q + 1$ possible values for o and p, and for m and n, respectively. Hence, when $o + p = q$, the number of feasible state is $(q+1)(|L_G|-q+1)$. Therefore, when $|L_G| \geq |G|$, the number

TABLE 2. State transition cases of each function or server.

Case	Type of event	State transition cases of each function or server
(a)	Trigger	A function in the active state fails and becomes waiting.
(b)	Trigger	A function in the failed and waiting state gets repaired and becomes active.
(c)	Trigger	A function in the active state fails and becomes being recovered.
(d)	Trigger	A function that is being recovered gets repaired and becomes active.
(e)	Induced	A function in the failed and waiting state becomes being recovered.
(f)	Induced	A function that is being recovered becomes waiting because of the server failure.
(g)	Trigger	A function that is recovered gets repaired and becomes active.
(h)	Induced	A function that is recovered becomes waiting because of the server failure.
(i)	Trigger	A function that is being recovered becomes recovered because of the completion of the recovery procedure.
(j)	Induced	A function that is recovered becomes being recovered because of the server failure.
(A)	Trigger	A server in the failed state gets repaired and waiting.
(B)	Trigger	A server that is active and waiting fails.
(C)	Trigger	A server in the failed state gets repaired and recovering a function.
(D)	Trigger	A server that is recovering a function fails.
(E)	Induced	A server that is active and waiting becomes recovering a function.
(F)	Induced	A server that is recovering a function stops recovering a function because of the function repairing.
(G)	Trigger	A server that has recovered a function fails.
(H)	Induced	A server that has recovered a function stops having recovered a function because of the function repairing.
(I)	Induced	A server that is recovering a function has recovered a function because of the completion of the recovery procedure.
(J)	Induced	A server that has recovered a function becomes recovering another function because of the function repairing.

TABLE 3. Situation of state transitions.

State transition	Trigger event	Induced event
1	(a)	
2	(b)	
3	(c)	(E)
4	(d)	(F)
5		(e)
6	(g)	(H)
7		(e) and (J)
8	(i)	(I)
9	(A)	
10	(B)	
11	(C)	(e)
12	(D)	(f)
13		(E)
14	(G)	(h)
15		(E) and (j)

TABLE 4. Feasible states incoming to and outgoing from state $\Gamma = (m, n, o, p, q)$.

Direction	Type	State transition	State	Transfer rate	Conditions
Outgoing	1	1	$(m-1, n+1, o, p, q)$	$m\lambda_{\mathrm{f}}$	$m \geq 1, (o+p=q \text{ or } q=0)$
	2	2 and 5	$(m+1, n-1, o, p, q)$	$(n+o)\mu_{\mathrm{f}}$	$n \geq 1,$
	3	3	$(m-1, n, o+1, p, q)$	$m\lambda_{\mathrm{f}}$	$m \geq 1, o+p+1 \leq q, q \geq 1$
	4	4	$(m+1, n, o-1, p, q)$	$o\mu_{\mathrm{f}}$	$n=0, o \geq 1$
	5	6	$(m+1, n, o, p-1, q)$	$p\mu_{\mathrm{f}}$	$n=0, p \geq 1$
	6	7	$(m+1, n-1, o+1, p-1, q)$	$p\mu_{\mathrm{f}}$	$n \geq 1, p \geq 1$
	7	8	$(m, n, o-1, p+1, q)$	$o\delta$	$o \geq 1$
	8	9	$(m, n, o, p, q+1)$	$(\lvert G\rvert - q)\mu_{\mathrm{s}}$	$n=0, q \leq \lvert G\rvert - 1$
	9	10	$(m, n, o, p, q-1)$	$(q-(o+p))\lambda_{\mathrm{s}}$	$n=0, o+p \leq q-1, q \geq 1$
	10	11	$(m, n-1, o+1, p, q+1)$	$(\lvert G\rvert - q)\mu_{\mathrm{s}}$	$n \geq 1, q \leq \lvert G\rvert - 1$
	11	12	$(m, n+1, o-1, p, q-1)$	$o\lambda_{\mathrm{s}}$	$o \geq 1, o+p=q, q \geq 1$
	12	13	$(m, n, o, p, q-1)$	$o\lambda_{\mathrm{s}}$	$o+p \leq q-1, o \geq 1$
	13	14	$(m, n+1, o, p-1, q-1)$	$p\lambda_{\mathrm{s}}$	$o+p=q, p \geq 1, q \geq 1$
	14	15	$(m, n, o+1, p-1, q-1)$	$p\lambda_{\mathrm{s}}$	$o+p \leq q-1, p \geq 1, q \geq 1$
Incoming	1'	1	$(m+1, n-1, o, p, q)$	$(m+1)\lambda_{\mathrm{f}}$	$n \geq 1$
	2'	2 and 5	$(m-1, n+1, o, p, q)$	$(n+1+o)\mu_{\mathrm{f}}$	$m \geq 1, (o+p=q \text{ or } q=0)$
	3'	3	$(m+1, n, o-1, p, q)$	$(m+1)\lambda_{\mathrm{f}}$	$n=0, o \geq 1$
	4'	4	$(m-1, n, o+1, p, q)$	$(o+1)\mu_{\mathrm{f}}$	$m \geq 1, o+p+1 \leq q, q \geq 1$
	5'	6	$(m-1, n, o, p+1, q)$	$(p+1)\mu_{\mathrm{f}}$	$m \geq 1, o+p+1 \leq q, q \geq 1$
	6'	7	$(m-1, n+1, o-1, p+1, q)$	$(p+1)\mu_{\mathrm{f}}$	$m \geq 1, o \geq 1, o+p=q$
	7'	8	$(m, n, o+1, p-1, q)$	$(o+1)\delta$	$p \geq 1$
	8'	9	$(m, n, o, p, q-1)$	$(\lvert G\rvert - (q-1))\mu_{\mathrm{s}}$	$n=0, o+p \leq q-1, q \geq 1$
	9'	10	$(m, n, o, p, q+1)$	$(q+1-(o+p))\lambda_{\mathrm{s}}$	$n=0, q \leq \lvert G\rvert - 1$
	10'	11	$(m, n+1, o-1, p, q-1)$	$(\lvert G\rvert - (q-1))\mu_{\mathrm{s}}$	$o \geq 1, o+p=q, q \geq 1$
	11'	12	$(m, n-1, o+1, p, q+1)$	$(o+1)\lambda_{\mathrm{s}}$	$n \geq 1, q \leq \lvert G\rvert - 1$
	12'	13	$(m, n, o, p, q+1)$	$o\lambda_{\mathrm{s}}$	$n=0, o \geq 1, q \leq \lvert G\rvert - 1$
	13'	14	$(m, n-1, o, p+1, q+1)$	$(p+1)\lambda_{\mathrm{s}}$	$n \geq 1, q \leq \lvert G\rvert - 1$
	14'	15	$(m, n, o-1, p+1, q+1)$	$(p+1)\lambda_{\mathrm{s}}$	$n=0, o \geq 1, q \leq \lvert G\rvert - 1$

of feasible states is $\sum_{q=0}^{|G|}(\frac{(q+1)q}{2}+(q+1)(|L_G|-q+1)) = \frac{1}{6}(|G|+1)(|G|+2)(3(|L_G|+1)-|G|) \coloneqq f_{\mathrm{num}}(G, L_G)$. When $|L_G| < |G|$, the number of feasible states is $\sum_{q=0}^{|L_G|}(\frac{(q+1)q}{2}+(q+1)(|L_G|-q+1)) = \frac{1}{6}(|L_G|+1)(|L_G|+2)(2|L_G|+3) = f_{\mathrm{num}}(L_G, L_G)$.

Next, we consider the number of feasible states in the condition of $|L_G| < q \leq |G|$. In this condition, $n=0$ since backup servers can start recovering at least another failed function. Therefore, $o+p = |L_G| - m$ and there are $|L_G| - m + 1$ possible values for o and p, and $|G| - |L_G|$ for q. As a result, when $|L_G| < q \leq |G|$, the number of feasible states is $\sum_{m=0}^{|L_G|}(|L_G|-m+1)(|G|-|L_G|) = \frac{|L_G|^2+3|L_G|+2}{2}(|G|-|L_G|) \coloneqq g_{\mathrm{num}}(G, L_G)$. Thus, the total number of feasible states is expressed by:

$$|U| = \begin{cases} f_{\mathrm{num}}(G, L_G) & (|L_G| \geq |G|) \\ f_{\mathrm{num}}(L_G, L_G) + g_{\mathrm{num}}(G, L_G) & (|L_G| < |G|), \end{cases} \quad (3)$$

where U denotes the set of all feasible states Γ.

B. SYSTEM STATE TRANSITION FOR Γ

There are 15 state transitions incoming to and outgoing from state Γ, respectively, according to Table 3. Table 4 describes the condition and transfer rate for each type of transition. The outgoing state transitions are named in order from 1 to 14, and the incoming state transitions are named in order from 1' to 14'. Each type corresponds to one state incoming to or outgoing from state Γ.

Types 1 through 8 involve state transitions triggered by functions, whereas types 9 through 14 involve those triggered by server state changes. For instance, examples of transition types 1, 9, and 11 are provided below.

- For type 1, an active function fails and goes to the failed and waiting state with the transfer rate of $m\lambda_{\mathrm{f}}$. Since there is at least one active function before the transition, we have a condition of $m \geq 1$. Since this state transition occurs if there is no backup server that can start the recovery procedure of the function, we have a condition of $o+p=q$ or $q=0$.
- For type 9, a backup server that is in the active and waiting state fails with the transfer rate of $(q-(o+p))\lambda_{\mathrm{s}}$ and goes to the failed state. Note that the number of backup servers that are in the active and waiting state before the transition is $q-(o+p)$. Since there is at least one backup server that is in the active and waiting state before the transition, we have a condition of $n=0$. Since the total number of functions that are either recovered or being recovered does not exceed q after the transition, we have a condition of $o+p \leq q-1$. Also, since there is at least one active backup server before the transition, we have a condition of $q \geq 1$.
- For type 11, a backup server that is recovering a function fails with the transfer rate of $o\lambda_{\mathrm{s}}$, and the function that is being recovered by the backup server stops being recovered and goes to the failed and waiting state. Note that the number of backup servers each of which is recovering a function before the transition is o. Since there is at least one function that is being recovered by a backup server before the transition, we have a condition of $o \geq 1$. Since there is no available backup server before the transition, $o+p=q$. Also, since there is at least one active backup server before the transition, we have a condition of $q \geq 1$.

The explanations for the system state transition incoming from state Γ mirror those for the outgoing transitions. Specifically, types 1 through 14 correspond directly to types 1' through 14'. For instance, descriptions of transition types 1', 9', and 11' are provided below.

- For type 1', an active function fails and goes to the failed and waiting state with the transfer rate of $(m+1)\lambda_{\mathrm{f}}$. This state transition occurs if there is no backup server that can start the recovery procedure of the function. Since there is at least one waiting function after the transition, we have a condition of $n \geq 1$.
- For type 9', a backup server that is in the active and waiting state fails with the transfer rate of $(q+1-(o+p))\lambda_{\mathrm{s}}$ and goes to the failed state. Note that the number of backup servers that are in the active and waiting state before the transition is $q+1-(o+p)$. Since there is at least one backup server that is in the active and waiting state before the transition, we have a condition of $n=0$. Also, since the number of active backup servers, $q+1$ does not exceed $|G|$, we have a condition of $q \leq |G|-1$.
- For type 11', a backup server that is recovering a function fails with the transfer rate of $(o+1)\lambda_{\mathrm{s}}$, and the function that is being recovered by the backup server stops being recovered and goes to the failed and waiting state. Note that the number of backup servers, each of which is recovering a function before the transition, is $o+1$. Since there is at least one waiting function after the transition, we have a condition of $n \geq 1$. Also, since the number of active backup servers, $q+1$ does not exceed $|G|$, we have a condition of $q \leq |G|-1$.

C. EXAMPLE OF STATE TRANSITIONS

Fig. 3 depicts an example of a state-transition diagram for a group consisting of two functions and two backup servers, i.e., $|F|=2$, $|S|=2$, and $|L_{\mathrm{G}}|=2$. Each node in Fig. 3 represents a state of the group, denoted by (m,n,o,p,q). We obtain the total number of states as $|U| = f_{\mathrm{num}}(G, L_G) = \frac{1}{6}(|G|+1)(|G|+2)(3(|L_G|+1)-|G|) = \frac{1}{6}(2+1)(2+2)(3(2+1)-2) = 14$, as described in Section IV-A. A directed edge in the diagram represents a state transition from one state to another, implying that such an edge represents a state transition in the *outgoing* category. State transitions in the *outgoing* category are classified into 14 types, as illustrated in Table 4. When transitioning from state A to state B, the transition observed from state A is in the *outgoing* category, while the transition observed from B is in the *incoming* category.

Each type of state transition is associated with a distinct edge representation using distinct colors. A blue edge represents a state transition triggered by function failure. An orange edge represents a state transition triggered by function repair. A black edge represents a state transition triggered by server failure. A green edge represents a state transition triggered by server repair. A purple edge represents a state transition triggered by function recovery.

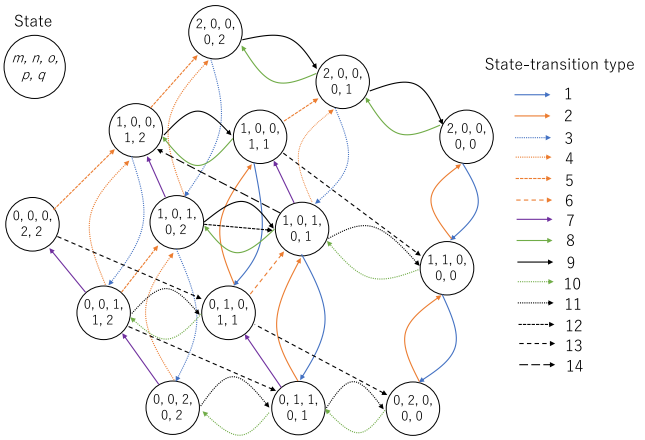


FIGURE 3. Example of state-transition diagram for a group consisting of two functions and two backup servers, i.e., $|F|=2$, $|G|=2$, and $|L_{\mathrm{G}}|=2$. Table IV explains each state-transition type.

Consider the state (0, 0, 0, 2, 2) placed at the left end of the diagram in Fig. 3 as an example. This state is outgoing to the states (1, 0, 0, 1, 2) and (0, 1, 0, 1, 1). These transitions are in the *outgoing* category for the state (0, 0, 0, 2, 2), each corresponding to types 4 and 13, respectively, as shown in Table 4. Type 4 represents a transition when a recovered function is repaired; the trigger event is (d), and the induced event is (F) in Table 3. Type 13 represents a transition when an active backup server with a recovered function fails; the trigger event is (G), and the induced event is (h) in Table 3.

The state (0, 0, 0, 2, 2) is incoming from (0, 0, 1, 1, 2). This transition is in the *incoming* category for the state (0, 0, 0, 2, 2); it corresponds to type 7' in Table 4. This transition is in the *outgoing* category for the state (0, 0, 1, 1, 2); it corresponds to type 7 in Table 4. Type 7 represents a transition when a function that is being recovered by a backup server gets recovered; the trigger event is (i), and the induced event is (I) in Table 3.

Types 9 and 12 have different state transitions but have the same ingress and egress states potentially leading to the occurrence of a directed multigraph. Consider the transition from (1, 0, 1, 0, 2) to (1, 0, 1, 0, 1) placed at the center of the diagram in Fig. 3 as an example. The transition has two types: types 9 and 12 in Table 4. Type 9 represents a transition when a backup server that is in the active and waiting state fails; the trigger event is (B) in Table 3, and no induced event occurs. Type 12 represents a transition when an active backup server that is recovering a function fails and the function that is being recovered by the backup server restarts being recovered by another available backup server; the trigger event is (D), and the induced event is (E) in Table 3. The distinction in the transition types is based on whether the failing backup server is in the active and waiting state or is recovering a function.

The usage of the multigraph expression, as shown in the example of transition types 9 and 12, depends on how we treat each state transition from the considered state to the same outgoing state in Table 4. Double edges between the

two states (1, 0, 1, 0, 2) and (1, 0, 1, 0, 1) in Fig. 3 exist since we distinguish state transitions 10 and 13 in Table 3 as separate transitions, even though the destinations of these transitions lead to the same state. State transitions 2 and 5 in Table 3 could be treated as different transition types, but we treat them as the same transition type 2 in Table 4 since they have simple conditions; for example, one edge between the two states (0, 1, 1, 0, 1) and (1, 0, 1, 0, 1) exist in Fig. 3 despite the potential categorization of the state transition as either 2 or 5 in Table 3. For state transitions 10 and 13 in Table 3, we treat them as different transition types in Table 4 since they have complex conditions.

D. EQUILIBRIUM STATES

In equilibrium, the total flows into state Γ is equal to the total flows out of state Γ. The equilibrium equation for Γ is expressed by:

$$\begin{aligned}
&h_1P(m+1,n-1,o,p,q)+h_2P(m-1,n+1,o,p,q)\\
&+h_3P(m+1,n,o-1,p,q)+h_4P(m-1,n,o+1,p,q)\\
&+h_5P(m-1,n,o,p+1,q)\\
&+h_6P(m-1,n+1,o-1,p+1,q)\\
&+h_7P(m,n,o+1,p-1,q)+h_8P(m,n,o,p,q-1)\\
&+h_9P(m,n,o,p,q+1)+h_{10}P(m,n+1,o-1,p,q-1)\\
&+h_{11}P(m,n-1,o+1,p,q+1)\\
&+h_{12}P(m,n,o,p,q+1)\\
&+h_{13}P(m,n-1,o,p+1,q+1)\\
&+h_{14}P(m,n,o-1,p+1,q+1)\\
&=\sum_{k=1'}^{14'}h_kP(m,n,o,p,q),\ \forall\Gamma\in U, \qquad (4)
\end{aligned}$$

where $h_k, k \in [1, 14]$ or $[1', 14']$, denotes the transfer rate of type k in Table 4 if the conditions of type k are satisfied and 0 otherwise.

The total probability of all states equals one, which is expressed by:

$$\sum_{\Gamma\in U}P(m,n,o,p,q)=1. \qquad (5)$$

We determine the probabilities of equilibrium states for each system state by solving a set of equations in (4) and (5).

The total number of feasible states denoted by $|U|$ can be obtained analytically, as described in Section IV-A. The equilibrium probabilities can be computed by using the Gauss-Seidel method, where the time complexity is $O(|U|^2)$ [31].

E. ESTIMATE UNAVAILABILITY OF FUNCTION

For state Γ, the number of unavailable functions is expressed by $n + o$. The average number of unavailable functions is expressed by $\sum_{\Gamma\in U}(n+o)P(m,n,o,p,q)$. The transition types that make the function unavailable are 1', 3', and 13'. For types $1'$ and $3'$, the function becomes unavailable with the rate of $m\lambda_f$. For type $13'$, the function becomes unavailable with the rate of $p\lambda_s$. Hence, per unit time, the average number of functions that become unavailable is expressed by $\sum_{\Gamma\in U}((m\lambda_f+p\lambda_s))P(m,n,o,p,q)$. Therefore, the average unavailable time of function f in a group is expressed by:

$$T_f^{\mathrm{u}}=\frac{\sum_{\Gamma\in U}(n+o)P(m,n,o,p,q)}{\sum_{\Gamma\in U}\left[(m\lambda_f+p\lambda_s)\right]P(m,n,o,p,q)}. \qquad (6)$$

Similarly, the average available time of function f in a group is expressed by:

$$T_f^{\mathrm{a}}=\frac{\sum_{\Gamma\in U}(m+p)P(m,n,o,p,q)}{\sum_{\Gamma\in U}[(n+o)\mu_f+o\delta)]P(m,n,o,p,q)}. \qquad (7)$$

Based on (6) and (7), the unavailability of a function f in a group can be computed by (1).

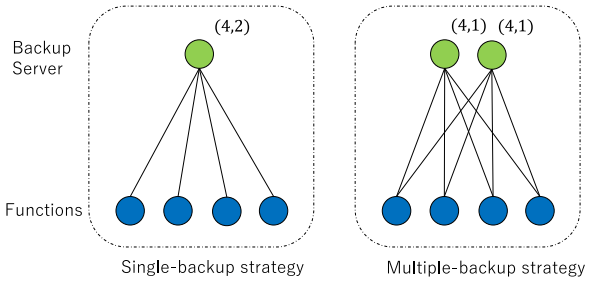
V. NUMERICAL RESULTS

We investigate how much the multiple-backup shared protection strategy reduces unavailability compared to the single-backup shared protection strategy. For simplicity, we refer to the single-backup shared protection strategy and the multiple-backup shared protection strategy as the single-backup strategy and the multiple-backup strategy, respectively, in this section if no confusion arises. We employ an analytical model introduced in [8] for the single-backup strategy and the proposed model for multiple-backup strategies, respectively. A backup server can recover up to r functions simultaneously. We call this r as *backup capacity*. In each of the single-backup and multiple-backup strategies, we call the sum of the backup capacity of each backup server as *backup resources* We conduct a comparison by equating the backup resources with different strategies. We investigate two cases of the single-backup and multiple-backup strategies, as shown in Fig. 4. Case 1 involves a single server in the single-backup strategy, and case 2 involves a single backup capacity in the single-backup strategy. Next, we investigate the time it takes to calculate unavailability with different numbers of functions and backup servers.

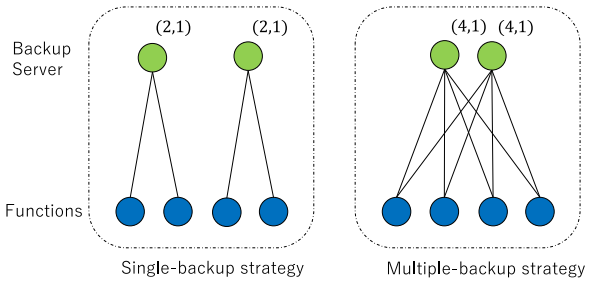
As per [8], the average time between two middlebox function failures is 10^4 seconds. When a function or server fails, it takes about 10^3 seconds to repair, and if it is on a backup server, recovery happens within 30 seconds, according to studies in [3]. We set λ_f^{-1}, μ_f^{-1}, μ_s^{-1}, δ^{-1} using these values. In addition, we set λ_s^{-1} to 10^4 seconds, which is the average time between two server failures. Our evaluation uses an Apple M1 3.2 GHz 8-core CPU with 16 GB memory.

A. CASE 1: COMPARISON OF UNAVAILABILITY BETWEEN SINGLE-BACKUP STRATEGY WITH A SINGLE SERVER AND MULTIPLE-BACKUP STRATEGY

In case 1, we compare the unavailability of a function between the single-backup strategy with a single server and



(a) Case 1



(b) Case 2

FIGURE 4. Examples of unavailability comparison between single-backup multiple-backup strategies. The left and right numbers in brackets attached to each server express the number of functions protected by each backup server and the backup capacity, respectively.

the multiple-backup strategy. In the single-backup strategy, each function is protected by a single backup server, and the backup server has multiple backup capacities. We calculate the unavailability in the single-backup strategy, based on the work in [8]. On the other side, in the multiple-backup strategy, each function is protected by multiple servers, and each server can recover only one function. Consequently, the backup capacity is set to one. The functions and servers together make up one connected component. Since the multiple-backup strategy can recover a function even if one backup server fails, we expect the unavailability of a function in the multiple-backup strategy to be lower than that in the single-backup strategy.

Fig. 4(a) shows an example of comparison, where symbols $(4, 2)$ and $(4, 1)$ are attached to one backup server in the single-backup strategy and each in the multiple-backup server strategy, respectively. The left and right numbers in brackets express the number of functions, denoted by f, protected by each backup server and the backup capacity, respectively. In the single-backup strategy, there is only one backup server, and its capacity is the same as the number of backup servers in the multiple-backup strategy. Thus, both are compared under the condition of having equal backup resources.

Fig. 5 shows the difference in unavailability between the single-backup and multiple-backup strategies. As the number of functions increases, the unavailability increases in both strategies. In the multiple-backup strategy with three backup servers, unavailability decreases by 6.99% to

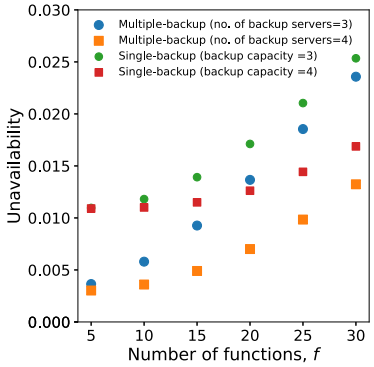


FIGURE 5. Comparison of unavailability between single-backup strategy with a single server and multiple-backup strategy in case 1.

66.7% compared to the single-backup strategy with a backup capacity of three, across a range of 5 to 30 functions. In the multiple-backup strategy with four backup servers, unavailability decreases by 21.6% to 72.3% compared to the single-backup strategy with a backup capacity of four, across a range of 5 to 30 functions.

These results mean the following. 1) When there are many more functions than backup resources, the difference in unavailability between the two strategies gets smaller. Thus, the multiple-backup strategy surpasses the single-backup strategy, except when the number of backup resources is significantly less than the number of functions. 2) When there are more backup resources, even if the ratio of backup resources to the number of functions stays the same, the difference in unavailability between the two strategies gets larger. For instance, when the backup resources are three and the number of functions is 15, the unavailability in the multiple-backup strategy decreases by 33.4% compared to that of the single-backup strategy. On the other hand, with four backup resources and 20 functions, the unavailability decreases by 44.4%. Thus, the superiority of the multiple-backup strategy over the single-backup strategy increases as the scale of the model becomes larger.

We conduct event-driven simulations. The assumptions and parameters that are utilized for the simulation are the same as those used for the analytical model. Our simulation provides unavailability with a 95% confidence interval that remains within 5% of each reported value.

Fig. 6 compares the unavailability obtained by the analytical model and simulation in case 1. We observe that the unavailability obtained by the analytical model and simulation are comparable; the difference between them is, at most, 3.4%.

B. CASE 2: COMPARISON OF UNAVAILABILITY BETWEEN SINGLE-BACKUP STRATEGY WITH A SINGLE BACKUP CAPACITY AND MULTIPLE-BACKUP STRATEGY

In case 2, we compare the unavailability of a function between the single-backup and multiple-backup strategies;

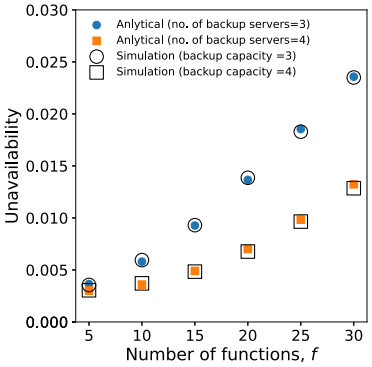


FIGURE 6. Unavailability of multiple-backup strategy obtained by analytical model and simulation in case 1.

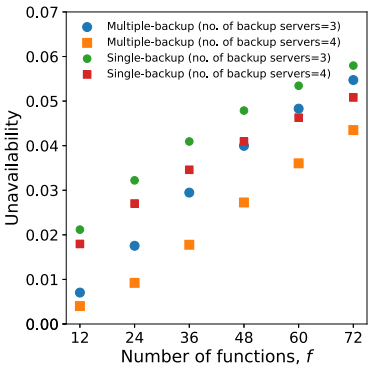


FIGURE 7. Comparison of unavailability between single-backup strategy with a single backup capacity and multiple-backup strategy in case 2.

each backup server in each model has a single backup capacity. Similar to case 1, we conduct a comparison by equating the backup resources.

Fig. 4(b) shows an example of comparison in case 2. In the single-backup strategy, there is only one backup capacity in each backup server, and the number of backup servers is the same in both strategies. Thus, both are compared under the condition of having equal backup resources. For simplicity, in the single-backup strategy, we ensure that the number of functions is the same for each component.

Fig. 7 shows the difference in unavailability between the multiple-backup strategy and the single-backup strategy. We compare the results of Fig. 5 and Fig. 7; with the current settings, we observe that a single-backup strategy with one backup server performs better than a single-backup strategy with only one backup capacity. We have similar observations to case 1. These unavailabilities gradually get closer as the number of functions becomes larger than the backup capacity. When there are more backup resources, the difference in unavailability between both strategies increases, even if the ratio of backup resources to the number of functions stays the same. Hence, when we compare both strategies, the multiple-backup strategy is also considered to be better than the single-backup strategy.

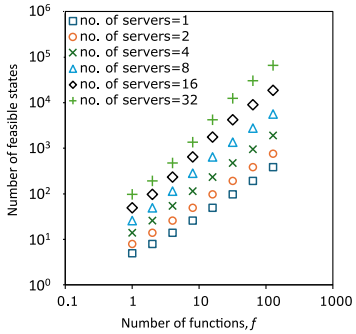


FIGURE 8. Comparison of number of feasible states dependent on number of functions for different numbers of backup servers.

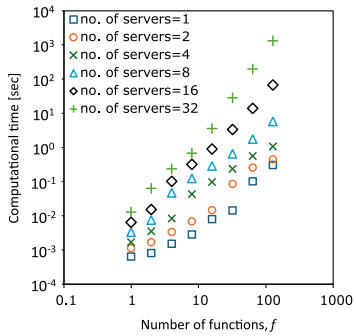


FIGURE 9. Comparison of computation time dependent on number of functions for different numbers of backup servers.

C. NUMBER OF FEASIBLE STATES AND COMPUTATION TIME WITH DIFFERENT NUMBERS OF FUNCTIONS AND BACKUP SERVERS

Fig. 8 shows the number of feasible states dependent on the number of functions with different numbers of servers. The number of feasible states grows with both the number of functions and backup servers. For example, when the number of functions is 64, the number of feasible states ranges from 10^2 to 10^5, depending on the number of backup servers.

Fig. 9 presents the computation time to compute the unavailability of multiple-backup models with different numbers of functions and backup servers, respectively. The time needed to compute the unavailability is at most 1.3×10^3 seconds in our examined cases. It increases as the number of functions and that of backup servers increases. The required computation time grows as the problem size increases. In our examined cases, the proposed analytical model is applicable for computing unavailability within a practical time.

As observed in Figs. 8 and 9, as the number of functions and that of backup servers increase, the number of feasible states and the computation time in the proposed model also increase. Consequently, the computation time may become impractically large. In such cases, an event-driven approach, such as Monte Carlo simulation, can be a practical alternative for evaluating unavailability. However, the proposed analytical model is still useful for validating simulation outcomes in

moderate-sized problems that it can handle. If the proposed analytical model is not applied, verifying the correctness of simulation outcomes would be difficult.

VI. BACKUP ALLOCATION

A. NATURE OF BACKUP ALLOCATION

We investigate the nature of backup allocation regarding unavailability. Firstly, groups possess the following properties.

Property 1: When the number of functions in a group is equal, the larger the number of backup servers is, the lower the unavailability.

Proof: Consider a group consisting of f functions and s backup servers.

Case 1: $f < s$

In this case, this group can recover a maximum of $s-k$ functions simultaneously, where $0 \leq k \leq s-1$, even if $s-f+k$ backup servers fail. Suppose that we add one backup server to this group. As a result, this group can recover a maximum of $s-k$ functions simultaneously even if $s+1-f+k$ backup servers fail. Therefore, the unavailability of this group decreases.

Case 2: $f \geq s$

In this case, this group can recover a maximum of k functions simultaneously, where $0 \leq k \leq s-1$, even if $s-k$ backup servers fail. Suppose that we add one backup server to this group. As a result, this group can recover a maximum of $k+1$ functions simultaneously even if $s-k$ backup servers fail. Therefore, the unavailability of this group decreases.

In either case, the unavailability of this group decreases. ■

Property 2: Suppose that two groups exist. The unavailability of a single group combining the two groups formed by connecting all their servers and functions is lower than the higher value of the unavailabilities of the two individual groups.

Proof: Suppose that when two groups are combined, the unavailability of the combined group does not increase. Let G_1 and G_2 represent each group, having s_1 and s_2 backup servers, respectively. Here, G_1 can recover a maximum of k functions simultaneously, where $1 \leq k \leq s_1-1$, even if s_1-k backup servers fail. Similarly, G_2 can recover a maximum of l functions simultaneously, where $1 \leq l \leq s_2-1$, even if s_2-l backup servers fail. When these two groups are integrated, the combined group can recover a maximum of m functions simultaneously, where $1 \leq m \leq s_1+s_2-1$, even if s_1+s_2-m backup servers fail. Consequently, the integrated group can recover more functions with the same number of backup server failures, which contradicts the assumption. Thus, the unavailability decreases when two groups are combined. ■

In light of this, we discuss what kind of allocation has low unavailability. The backup server can protect c functions simultaneously. We call this c as *service capacity*. Firstly, assume that the service capacity of the backup server is equal to the number of all functions. In this case, the optimal allocation is the one that connects all functions to each backup server.

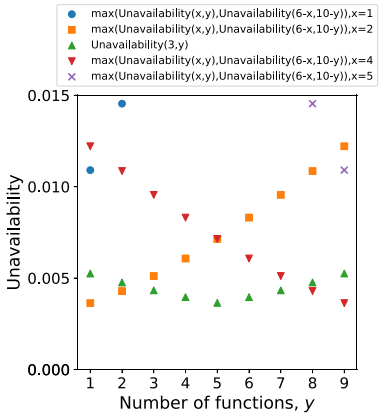


FIGURE 10. Graph of unavailability when considering an allocation of two groups.

Property 3: When the service capacity is equal to the number of all functions, the optimal allocation is the one that connects all functions to each backup server.

Proof: Suppose that any allocation consisting of a single group is not an optimal allocation; then allocations comprising m (≥ 2) groups become the optimal allocation. Consider connecting all functions and servers of two groups within this allocation. In this case, the unavailability after the connection is always less than or equal to that of the original allocation, which contradicts the assumption. Therefore, an allocation of $m \geq 2$ groups is not optimal. ■

Next, consider the case where the service capacity is less than $|F|$, which does not satisfy Property 2. In this case, it is not possible to take an allocation that connects all backup servers and functions. At this point, the optimal allocation is classified into two categories.

The first category is to allocate a group with many backup servers for a relatively small number of functions and allocate the remaining functions into a group that connects all of them. We call this allocation an *unbalanced allocation*. This allocation often becomes optimal when the service capacity is close to $|F|$.

The second category is an allocation that divides groups as evenly as possible. We call this allocation a *balanced allocation*. This allocation often becomes optimal when the service capacity is small and the creation of multiple groups is necessary.

We specifically investigate an allocation scenario with six servers and ten functions under the condition that the service capacity is nine. Fig. 10 illustrates the graph of unavailability when considering an allocation of two groups. In Fig. 10, unavailability (x,y) represents the unavailability of a group composed of x servers and y functions. Fig. 10 shows only the vicinity of the minimum unavailability. By examining Fig. 10, one can determine which allocation minimizes the group's unavailability in this scenario, which has six servers and ten functions.

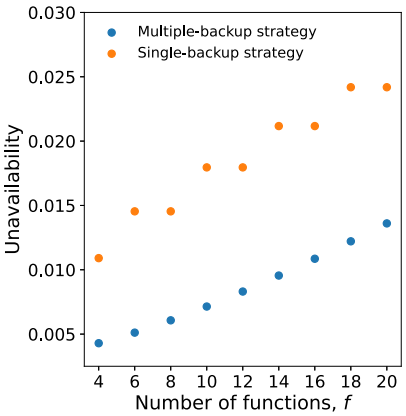


FIGURE 11. Comparison of unavailability between multiple-backup and single-backup strategies for each best allocation when the number of backup servers is four and the service capacity is $\frac{f}{2}$.

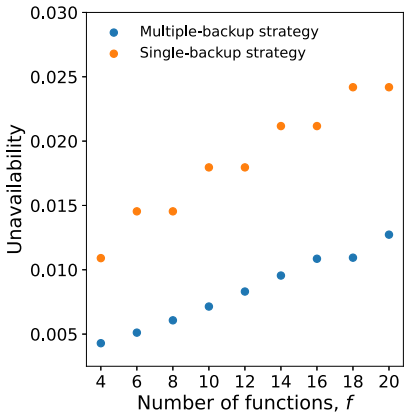


FIGURE 12. Comparison of unavailability between multiple-backup and single-backup strategies for each best allocation when the number of backup servers is four and the service capacity is $f - 1$.

Fig. 10 shows that the backup allocation of (the number of servers, the number of functions) = (2, 1) and (4, 8) and that of (3, 5) and (3, 5) are candidates for the minimum unavailability. In this context, the backup allocation of (2, 1) and (4, 9) corresponds to the unbalanced allocation, while the backup allocation of (3, 5) and (3, 5) corresponds to the balanced allocation. In the current parameter setting, the backup allocation of (2, 1) and (4, 9) attains the minimum unavailability. However, it is worth noting that parameter configurations exist where the backup allocation of (3, 5) and (3, 5) achieves the minimum unavailability.

B. COMPARISON BETWEEN BACKUP ALLOCATION STRATEGIES

We conduct a comparison between the multiple-backup strategy and the single-backup strategy; we investigate the unavailability of the best allocation corresponding to each strategy. We calculate the unavailability for all feasible allocations in each strategy and designate the one with the smallest unavailability as the best allocation. In other words, we handle the UASBA problem described in Section III-D for both strategies. In this comparison, for both strategies, we set the maximum number of backup servers as s, the number of functions as f, the backup capacity as 1, and the service capacity as c.

1) CASE 1

We compare the unavailability of the best allocation corresponding to each strategy when the number of backup servers s is four and the service capacity c is $\frac{f}{2}$. Fig. 11 compares the unavailability between multiple-backup and single-backup strategies for each best allocation in case 1. The multiple-backup strategy reduces the unavailability of a function by up to 64.8% compared to the single-backup strategy.

2) CASE 2

We compare the unavailability of the best allocation corresponding to each strategy when the number of backup servers s is four and the service capacity is $f - 1$, which is the highest capacity not meeting Property 3. Fig. 12 compares the unavailability between multiple-backup and single-backup strategies for each best allocation in case 2. The multiple-backup strategy reduces the unavailability of a function by up to 64.8% compared to the single-backup strategy.

When we compare Fig. 12 in case 2 with Fig. 11 in case 1, the results only differ when the number of functions, f, is 18. We can explain this as follows. The optimal backup allocation in case 2 for (the number of servers, the number of functions) = (4, 18) is (1, 1) and (3, 17), where the higher unavailability of the two groups is 0.01094. Meanwhile, the unavailability for groups (1, 1), (3, 17), and (2, 9) is 0.01090, 0.009268, and 0.01221, respectively. Consequently, the backup allocation of (1, 1) and (3, 17) exhibits lower unavailability than the backup allocation of (2, 9) and (2, 9), whose unavailability is 0.01221. This is the case when an unbalanced allocation is better than a balanced allocation, as we discussed in the last paragraph of Section VI-A.

For other numbers of functions, i.e., $f \neq 18$, the backup allocation and unavailability in both Fig. 11 and 12 are the same, whereas the service capacity is different. Thus, the multiple-backup strategy still works well even if the service capacity is not so large.

3) CASE 3

We compare the unavailability of the best allocation corresponding to each strategy when the number of backup servers s is three and the service capacity is $f - 1$. Fig. 13 compares the unavailability between multiple-backup and single-backup strategies for each best allocation in case 3. The multiple-backup strategy reduces the unavailability of a function by up to 42.3% compared to the single-backup strategy.

Fig. 13 does not show much improvement in multiple-backup allocation compared to Fig. 12 since splitting the backup servers evenly and balancing the backup allocation is

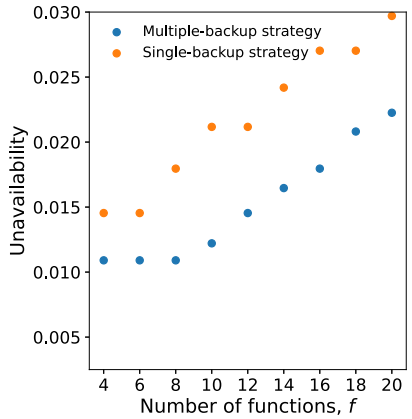


FIGURE 13. Comparison of unavailability between multiple-backup and single-backup strategies for each best allocation when the number of backup servers is three and the service capacity is $f - 1$.

impossible. In addition, when we have fewer backup servers, unbalanced allocation does not work well enough. This is because all backup allocations end up with one and two backup servers respectively, creating such backup allocation that mixes these two allocation patterns.

In Fig. 13, when the number of functions ranges from four to eight, the unavailability is consistently 0.01090. This is because the best allocation includes the group of (1,1), whose unavailability is 0.01090, which is higher than any other group. In a situation where balanced allocation is not feasible, there exists a potential for performance degradation, especially when the number of functions is relatively small, which corresponds to the region of $f \in [2, 6]$ in fig10.

Hence, the multiple-backup strategy works better when we can evenly split the servers or when there are enough backup servers. In this case, unbalanced backup allocation can be effective.

VII. CONCLUSION

A. SUMMARY

This paper proposed an analytical model that assesses the unavailability of middlebox functions for multiple-backup shared protection. We analyzed the state transitions to make equilibrium-state equations. We solve the equilibrium-state equations by applying a Markov chain to obtain an unavailability of function. Using the proposed analytical model, we compared the unavailability of a function between the multiple-backup and single-backup shared protection. The numerical results showed that the multiple-backup shared protection strategy reduces the unavailability of a function by up to 72.3% compared to the single-backup shared protection strategy in our examined cases. We also investigated the nature of backup allocation and compared the unavailability of the best allocation corresponding to each strategy. The best multiple-backup allocation reduces the unavailability of a function by up to 64.8% compared to the best single-backup allocation in our examined cases.

B. FUTURE WORK

The proposed analytical model adopted basic assumptions. This modeling simplifies the backup capacity by assuming that a backup server handles at most one function. The model does not consider situations where a backup server can recover multiple functions simultaneously, where the backup capacity depends on factors other than just the number of functions, or when parameters such as a failure rate vary depending on different backup servers or functions.

Our model is in its early stages, but it has the potential to be extended to accommodate these scenarios. Addressing these issues requires increasing the number of feasible states and state transitions, making the model more complex. Studies [26], [32], [33] presented analytical models to estimate the blocking probability of lightpath requests in elastic optical networks [34], [35]. Their algorithms effectively generate feasible states for spectrum allocation in an optical link. Inspired by these approaches, we can adopt similar techniques to efficiently manage feasible states and state transitions in our extended model, which will accommodate more complex scenarios. These challenges are left as subjects for future research.

In this paper, we assume that function failures occur independently. Specifically, different functions are allocated to different physical machines, and each function failure is independent of others. However, in reality, multiple functions may be deployed on the same physical machine, and a failure of that machine would lead to correlated failures of all functions hosted on it. Study [14] analyzed function unavailability considering physical machine failures under a single-backup shared protection strategy. However, no studies have analyzed function unavailability under a multiple-backup shared protection strategy when functions are colocated on a physical machine that can fail. Extending our analytical model to incorporate correlated failures in a multiple-backup shared protection scenario remains a crucial direction for future research.

ACKNOWLEDGMENT

The authors are grateful to Nozomi Kita for helpful discussions.

REFERENCES

[1] N. Wakuda, K. Nozomi, R. Shiraki, and E. Oki, "Analytical model for assessing unavailability in middleboxes with multiple backup servers under shared protection," in *Proc. Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2024, pp. 1–8.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function Virtualization: State-of-the-art and research challenges," in *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[3] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters," in *Proc. Int. Meas. Conf. (IMC)*, 2013, pp. 9–22.

[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, Oct. 2018.

[5] F. He, T. Sato, and E. Oki, "Optimization model for backup resource allocation in middleboxes," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, Tokyo, Japan, 2018, pp. 1–6.

[6] F. He, T. Sato, and E. Oki, "Optimization model for backup resource allocation in Middleboxes with importance," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1742–1755, Aug. 2019.
[7] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2759–2772, Oct. 2017.
[8] F. He and E. Oki, "Unavailability-aware shared virtual backup allocation for middleboxes: A queueing approach," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2388–2404, Jun. 2021.
[9] R. Fujita, F. He, and E. Oki, "Analytical model of middlebox unavailability under shared protection allowing multiple backups," *IEICE Trans. Commun.*, vol. E104-B, no. 9, pp. 1147–1158, Sep. 2021.
[10] K. Yokouchi, F. He, and E. Oki, "Backup resource allocation of virtual machines with two-stage probabilistic protection," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 4, pp. 5085–5102, Dec. 2023.
[11] K. Yokouchi and E. Oki, "Multiple-backup resource allocation for virtual machines with probabilistic protection in cloud," in *Proc. IEEE Commun. Qual. Rel. Workshop*, Oct. 2023, pp. 7–12.
[12] N. Kita, F. He, and E. Oki, "Unavailability-aware backup allocation model for middleboxes with two-stage shared protection," in *Proc. Proc. IEEE 8th Int. Conf. Netw. Softw. (NetSoft)*, 2022, pp. 384–392.
[13] N. Kita, H. Fujun, and E. Oki, "Unavailability-aware backup allocation model based on two-stage shared protection for middleboxes," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 357–372, Feb. 2014.
[14] N. Kita and E. Oki, "Unavailability-aware allocation of backup resources considering failures of virtual and physical machines," *Comput. Netw.*, vol. 239, Feb. 2024, Art. no. 110141.
[15] H. A. Alameddine, S. Ayoubi, and C. Assi, "An efficient survivable design with bandwidth guarantees for multi-tenant cloud networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 357–372, Jun. 2017.
[16] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-aware backup allocation for a chain of virtual network functions," in *Proc. IEEE INFOCOM*, 2019, pp. 1918–1926.
[17] D. Zheng, H. Fang, S. Cao, Y. Zhong, and X. Cao, "Towards resources optimization in deploying service function chains with shared protection," *Comput. Netw.*, vol. 248, Jun. 2024, Art. no. 110494.
[18] D. Zheng, G. Shen, Y. Li, X. Cao, and B. Mukherjee, "Service function chaining and embedding with heterogeneous faults tolerance in edge networks," *IEEE Trans Netw. Service Manag.*, vol. 20, no. 3, pp. 2157–2171, Sep. 2023.
[19] C. Peng, D. Zheng, Y. Zhong, and X. Cao, "Off-site protection against service function forwarder failures in NFV," *Comput. Netw.*, vol. 221, Feb. 2023, Art. no. 109510.
[20] D. Zheng, G. Shen, B. Chen, C. Peng, X. Cao, and B. Mukherjee, "Embedding service function chains with dedicated protection in edge networks," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 365–370.
[21] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proc. 4th Annu. Symp. Cloud Comput.*, 2013, pp. 1–15.
[22] J. Sherry et al., "Rollback-recovery for middleboxes," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 227–240, Aug. 2015.
[23] M. Guida, M. Longo, and F. Postiglione, "Performance evaluation of IMS-based core networks in presence of failures," in *Proc. IEEE Global Telecommun. Conf.*, 2010, pp. 1–5.
[24] M. Di Mauro, M. Longo, and F. Postiglione, "Availability evaluation of multi-tenant service function chaining infrastructures by multidimensional universal generating function," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1320–1332, Sep./Oct. 2021.
[25] K. Arakawa and E. Oki, "Availability-aware virtual network function placement based on multidimensional universal generating functions," *Int. J. Netw. Manag.*, vol. 34, Mar. 2024, Art. no. e2252.
[26] I. Ahmed, R. K. Rai, M. Maity, E. Oki, and B. C. Chatterjee, "AnalyticalBP: Analytical model for blocking probabilities considering crosstalk-avoided approach in spectrally-spatially elastic optical networks," *IEEE Trans. Commun.*, vol. 72, no. 3, pp. 1487–1501, Mar. 2024.
[27] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, Aug. 2008.
[28] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 4, pp. 337–350, Oct.–Dec. 2010.
[29] P. Garraghan, P. Townend, and J. Xu, "An empirical failure-analysis of a large-scale cloud computing environment," in *Proc. IEEE 15th Int. Symp. High-Asaance Syst. Eng.*, 2014, pp. 113–120.
[30] B. Tola, Y. Jiang, and B. E. Helvik, "Failure process characteristics of cloud-enabled services," in *Proc. 9th Int. Workshop Resilient Netw. Design Model (RNDM)*, 2017, pp. 1–7.
[31] A. Björck, *Numerical Methods in Matrix Computations*. Cham, Switzerland: Springer, 2015.
[32] I. Ahmed, E. Oki, and B. C. Chatterjee, "AnDefrag: Analytical model for blocking probabilities considering defragmentation in spectrally-spatially elastic optical networks," *IEEE Trans. Netw.*, early access, Feb. 26, 2025, doi: 10.1109/TON.2025.3538715.
[33] I. Ahmed, R. K. Rai, E. Oki, and B. C. Chatterjee, "AnalyticalDF: Analytical model for blocking probabilities considering spectrum defragmentation in spectrally-spatially elastic optical networks," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 1960–1969.
[34] B. C. Chatterjee, N. Sarma, and E. Oki, "Routing and spectrum allocation in elastic optical networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1776–1800, 3rd Quart., 2015.
[35] B. C. Chatterjee, S. Ba, and E. Oki, "Fragmentation problems and management approaches in elastic optical networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 183–210, 1st Quart., 2018.

NAOHIDE WAKUDA (Graduate Student Member, IEEE) received the B.E. degree from Kyoto University, Kyoto, Japan, in 2023, where he is currently pursuing the M.E. degree with the Graduate School of Informatics. His research interests include analyzing the performance of network systems.

RYUTA SHIRAKI (Member, IEEE) received the Ph.D. degree in engineering from Nagoya University, Aichi, Japan, in 2023. He is currently an Assistant Professor with Kyoto University, Kyoto, Japan. His research interests broadly lie in optical networks. Recently, his work has focused on digital signal processing, network architecture, network design, and artificial intelligence.

EIJI OKI (Fellow, IEEE) is a Professor with Kyoto University, Kyoto, Japan. He was with Nippon Telegraph and Telephone Corporation Laboratories, Tokyo, from 1993 to 2008, and The University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar with Polytechnic University, Brooklyn, NY, USA. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks.