Shared backup allocation model of middlebox based on workload-dependent failure rate

Han Zhang, Fujun He, and Eiji Oki
Graduate School of Informatics, Kyoto University, Kyoto, Japan

Abstract—This paper proposes a shared backup allocation model of middlebox with considering the failure, repair, recovery rates of each element and the workload of backup server. Multiple functions can share the backup resources on a backup server. By analyzing the process of failure, repair and recovery, we can get four possible states for each function, and the system state transition. Through the queuing approach, we can compute the probability of each function being available or unavailable in a certain assignment, and obtain the unavailability of each function. The proposed model aims to find an assignment to minimize the maximum unavailability among functions. We solve this problem by the simulated annealing algorithm. We evaluate and compare performance of proposed and baseline models under different experimental conditions. Based on the results, we can observe that, compared to the baseline model, the proposed model reduces the average maximum unavailability by an average of 15% in our examined cases.

Index Terms—Workload-dependent, queuing theory, network function virtualization, backup resource allocation

I. INTRODUCTION

Middleboxes play an important role in the network, by providing a variety of functions, such as deep packet inspection (DPI), network address translators (NAT), load balancing, firewall and so on. Traditionally, a middlebox runs on a physical dedicated server, and provides one single function. On the other hand, by implementing the network function virtualization technology [?], [?], middleboxes can be utilized as software on servers, which allows hardware resources to be used more efficiently.

Middleboxes can fail due to various reasons. The failure can cause interruption of service or security problems, which leads to degradation of service reliability [?]. Therefore, the backup for middleboxes is necessary.

Active-standby is a common approach: a standby instance is prepared for each active instance, if the backup server doesn't fail, it can become active and provide the services in place of the middlebox when the middlebox fails [?], [?]. However, this approach may cause overhead and waste in terms of the large number of backup servers.

On the contrary, in the shared backup, the number of functions that a backup server can protect is larger than the number of functions that it can recover at the same time. In this backup method, the resources of the backup servers are able to be used more efficiently, which reduces the required backup capacity. Basically, with shared backup, we do not need as many backup servers as the number of functions to achieve the acceptable availability.

should I write something here?

The work in [?] analyzes a basic scenario that one backup server can protect two functions and recover one function at the same time; it aims to find the optimal assignments to maximize the survivability of functions. In the model of [?], backup servers are assumed never to fail, and the time required to recover functions was not taken into account. The work in [?] follows the basic assumptions of [?], and introduces a 1:2 protection model, which means that a backup server can recover half of the functions protected by it at the same time.

The works in [?], [?] focus on the workload-dependent failure probability and try to find an assignment to minimize the maximum expected unavailable time (MEUT) with the workload-dependent failure probability and different backup strategies; the backup server can fail in this case, and the failure probability is affected by the existing workload on the backup server. For a server which can recover multiple functions at the same time, if a part of the recovery capacity has been taken up, it can have the same or a higher failure probability than other backup servers that have recovered less functions.

The works in [?], [?] introduce the method with queuing to analyze the unavailability of middleboxes and find an assignment to minimize the maximum unavailability among functions; as the prerequisites, the backup servers can fail, the failed middleboxes cannot be recovered immediately but need time, and the average failure rate of element is given as a constant. It comprehensively takes into account more factors including the failure of backup servers and the recovery time of functions and backup servers, which were not considered by the previous works. It does not consider the influence of workload on the failure probability. In the state transition, the failure probability of the backup server remains the same regardless of the workload on the backup server in the certain state. Therefore, when we analyze the unavailability and try to get the optimal assignment according to the given constant failure probability and other parameters, the adopted constant failure probability may be conservatively set to the value at the highest workload under workload-dependent situation. In this situation, there can be such a backup server: because the functions protected by it have higher failure probabilities than those protected by other backup servers, it is allocated fewer functions while its recovery capacity is higher, which leads to insufficient utilization for its capacity. From the point of view of workload-dependent failure probability, this backup server actually has a lower failure probability, so the unavailability of the functions protected by it is actually lower. Therefore, a part of the functions of the connected component with the maximum unavailability in the current assignment can be allocated to the backup server to reduce the overall maximum unavailability. The resulting assignment may not be optimal without considering the workload. So, it is necessary to integrate the workload-dependent failure probability into queuing method. There is no study to discuss this scenario.

Two questions arise: 1) How can we obtain the unavailability when the failure probability is not a constant? 2) How much the unavailability can be reduced compared to previous work if considering the workload-dependent failure probability?

To answer these questions, this paper proposes a model by analyzing the state transition of functions and backup servers considering the workload-depending failure probability. In our model, resources of backup servers can be shared by different functions. When an active function fails, if the backup server protecting it is active and has idle recovery capacity, the function start to be recovered by the backup server; otherwise, the function enters to the waiting state. After the recovering procedure is over, the function works on the backup server. When the repair of the failed function is completed, it becomes active. We consider that, as long as the recovery resources of the backup server are occupied, that is, functions are being recovered by the backup server or functions have been recovered by the backup server, the workload of the backup server increases, which leads to increasing of failure probability of backup servers. Once the backup server fails, functions which are being recovered or have been recovered enters to the waiting state. The proposed model aims to find the optimal assignment to minimize the maximum unavailability among functions. According to the queuing approach, we can get the state transition by considering the failure, repair, waiting and recovering procedures. By solving the multiple linear equations generated by the state transition, the steady probability of each state can be obtained. Using the steady probability, we can get the average available time and the average unavailable time of the assignment and compute the unavailability. The result shows that the proposed model reduces the average maximum unavailability by an average of 15% compared to the baseline model in our experiments.

The rest of the paper is organized as follows. Section II describes the proposed model. Section III demonstrates the method for analyzing unavailability. Section IV compares the performance of the proposed and baseline models. Section V concludes this paper.

II. MODEL AND PROBLEM DEFINITION

A. MODEL

Let F and S denote a set of functions and a set of backup servers, respectively. Every element in $W = F \cup S$ is either in an active state or failed state. One backup server can protect multiple functions; one function can only be protected by one server. Table I summarizes the frequently used notations.

TABLE I
LIST OF FREQUENTLY USED NOTATIONS

Notations	Meaning
c_j	Given parameter indicating the number of functions that can be protected by backup server $j \in S$
r_j	Given parameter indicating the number of functions that can be recovered by backup server $j \in S$ at the same time
λ_h	Given parameter indicating average failure rate of function with class $h \in H$
$\lambda(\omega)$	Average failure rate of backup server under workload ω
μ_h	Given parameter indicating average repair rate of the function and backup server with class $h \in H$
δ_j^{-1}	Given parameter indicating average recovery time on server $j \in S$
x_{ij}	Binary variable indicating whether backup server $j \in S$ is assigned to protect function $i \in F$
L_j	Set of functions protected by backup server $j \in S$
m_h	Number of functions in active state with class $h \in H$
n_h	Number of failed functions in waiting procedure with class $h \in H$
o_h	Number of failed functions in recovery procedure with class $h \in H$
p_h	Number of failed functions after completing recovery procedure with class $h \in H$
q	Number of backup servers in active state in group

We assume that each active function in F encounters the failure independently to other elements in W. We assume that the failure events occur at an function based on a Poisson process [?]. Different functions can have different failure rates due to various factors such as different running times; thus, we can classify the functions in F to several classes, in each of which the functions have the same failure rate [?]. The set of all classes is denoted by H; the failure rate of an function in class $h \in H$ is λ_h.

On the other side, each active backup server in S encounters the failure independently to other backup servers in S; it can be affected by the functions to protect. To protect functions against the failure, a backup server j is assigned to protect a set of functions L_j where the number of functions does not exceed its protection capacity c_j. r_j denotes the recovery capacity of backup server j. When a function in L_j fails, the backup server j starts to recover it if the number of failed functions is less than r_j, which means the number of functions that backup server j can recover at the same time. When none of the functions in L_j fails, server j only works on the information synchronization and has the lowest workload. As the number of failed functions increases, the workload on the backup server j increases, resulting in an increase in the failure rate of itself. When the number of failed functions equals or exceeds the recovery capacity of r_j, the workload and the failure rate of the backup server j reach the maximum at this time. We can regard the failure rate of the backup server as a function of workload; the workload can be regarded as the ratio of the number of failed function to the recovery capacity. A function that describes the workload-dependent failure rate is given by statistics of actual situation.

When an element in W fails, the repair procedure starts. The repair procedure can be different due to the different problems, such as replacing the parts on the broken physical machine, correcting the wrong configuration, and restarting the operating system. In order to simplify the model, we consider that the time to complete the repair of the element in class $h \in H$ follows an exponential distribution with the average value of μ_h^{-1} unit time.

B. FEASIBLE STATES AND STATE TRANSITION

There are four possible states for a function: active, failed and being recovered on a backup server (being recovered for short), failed and recovered on backup server (recovered for short), and failed and waiting (waiting for short). The first is the active state; one situation is that the function never experience a failure and stays in the active state; the other situation is that the function fails and the repair procedure for it is over, and then it turns to the active state. Depending on the remaining recovery capacity of the backup server, the function in the active state can fail and moves to the failed and being recovered state.

The second is the failed and being recovered state. If the backup server is active and has remaining recovery capacity, the backup server starts the recovery procedure for a failed function before its repair procedure is over; the failed function can move to the failed and being recovered state from the waiting state or the active state. The recovery procedures need time; we assume that the recovery time for the failed function in backup server j follows an exponential distribution with the average value of δ_j^{-1}.

The third is the failed and recovered state. Once the recovery procedure is over, the function runs on the backup server before the repair procedure is completed; the function moves to the failed and recovered state from the being recovered state.

The fourth is the waiting state. If an active function fails and the backup server has no remaining recovery capacity, it moves to the waiting state. If the backup server fails, the functions in the being recovered state and recovered state move to the waiting state immediately.

Fig. 1. State transition for each function

C. UNAVAILABILITY

For function $i \in F$, there exist average availability T_i^{AT} and average unavailability T_i^{UT}. The unavailability Q_i can be defined as the ratio of the average unavailability T_i^{UT} to the sum of average unavailability T_i^{UT} and the average availability T_i^{AT} [?], [?]:

$$Q_i = \frac{T_i^{\mathrm{UT}}}{T_i^{\mathrm{UT}} + T_i^{\mathrm{AT}}} \tag{1}$$

D. PROBLEM DEFINITION

In one assignment, there exists an maximum unavailability among all functions; in different assignments, maximum unavailability is usually different. We define the problem as follows:

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

III. ANALYSES FOR UNAVAILABILITY

Let $L_j = \{i | i \in F : x_{ij} = 1\}$ represent the set of functions protected by backup server $j \in S$. Backup server j can protect multiple functions. Because the functions protected by j cannot be recovered by backup servers other than j, L_j and j form a connected component. The number of connected components equals the number of backup servers. We analyze the unavailability of functions in each connected component respectively. Let $L_j^h \subseteq L_j$ denotes the set of all function with class $h \in H$ in L_j. In each connected component, because the functions with the same class have the same failure rate and average repair time, they also have the same unavailability.

Let a set of numbers, (m_h, n_h, o_h, p_h), represent the state of the functions with the class$h$ in the same connected component containing backup server j, where m_h, n_h, o_h and p_h represent the numbers of the functions with class h which are active, waiting, being recovered, and recovered, respectively. Let $q = 1$ when backup server j is active; otherwise, $q = 0$. The state of the whole connected component including backup server j and all L_j^h with the different classes is represented as $(m_1, n_1, o_1, p_1, ..., m_H, n_H, o_H, p_H, j)$. We start from the basic case, $|H| = 1$, where the state of one single connected component is expressed by (m, n, o, p, q).

Since the number of functions in the being recovered and recovered states can not exceed the recovery capacity r_j of backup server j, there exists $\sum_{h \in H}(o_h + p_h) \leq r_j$. The number of functions can not exceed the protection capacity c_j, so there exists $m + n + o + p \leq c_j$.

Fig. 2. State transition for (m, n, o, p, q)

A. SYSTEM STATE TRANSITION

Figure 2 shows the state transition for (m, n, o, p, q). There are nine types of states incoming to (m, n, o, p, q), and we number them from 1 to 9; accordingly, there are nine types of states outgoing from (m, n, o, p, q), and we number them from 10 to 18.

We can explain the state transition according to the nine types of state incoming to (m, n, o, p, q).

For types 1-4, a function goes to the active state when the repair procedure of the function is finished.

1) For type 1, there are two situations. The first is that a function in the waiting state goes to the active state with the transfer rate of $(n+1)\mu$. The second situation is that a function in the recovered state goes to the active state with the transfer rate of $o\mu$, and another function in the waiting state goes to the being recovered state at the same time. Since at least one function is in the waiting state before the transition, the number of failed functions exceeds the recovery capacity of j before the transition; there exists $n+o+p+1 > r_j$.
2) For type 2, a function in the being recovered state goes to the active state with the transfer rate of $(o+1)\mu$. Since at least one function is in the being recovered state before the transition, the backup server has to be active, which leads to $q = 1$. Obviously, there's no function in the waiting state in this case, so there are conditions of $n = 0$ and $n+o+p+1 \leq r_j$.
3) For type 3, a function in the recovered state goes to the active state with the transfer rate of $(p+1)\mu$. Similar to type 2, there are conditions of $q = 1$, $n = 0$, and $n+o+p+1 \leq r_j$.
4) For type 4, a function in the recovered state goes to the active state with the transfer rate of $(p+1)\mu$, and another function in the waiting state goes to being recovered state at the same time. Similar to type 1, there exist $n > 0$ and $n+o+p+1 > r_j$. Similar to types 2 and 3, there is a condition of $q = 1$.

For type 5, backup server j goes to the active state when the repair procedure is finished with the transfer rate of μ, so there exists $q-1 = 0$. All the functions in the waiting state go to the being recovered state at the same time, which means $p = 0$.

For types 6 and 7, a function in the active state fails.

1) For type 6, a function fails and goes to the waiting state with the transfer rate of $(m+1)\lambda$.
2) For type 7, a function fails and goes to the being recovered state with the transfer rate of $(m+1)\lambda$. Backup server j is active and has idle recovery capacity before the transition, therefore, there are conditions of $n = 0$, $q = 1$, and $n+o+p \leq r_j$.

For type 8, backup server j fails with the transfer rate of $\lambda(\omega)$, and all functions in the being recovered state and recovered state go to the waiting state. There exists $o = p = 0$ and $q = 0$.

For type 9, a function in the being recovered goes to the recovered state with the transfer rate of $(o+1)\delta_j$when its recovery procedure is finished. Since the backup server has to be active and there is at least one function in the being recovered state before the transition, there exist $q = 1$ and $o+1 > 0$.

The transitions outgoing from (m, n, o, p, q) to types 10-18 are similar to above. Table II shows the transfer rate and conditions for each state transition incoming to state (m, n, o, p, q). Table III shows the transfer rate and conditions for each state transition outgoing from state (m, n, o, p, q).

B. UNAVAILABILITY OF FUNCTION

Let U_j denote a set of all states in the connected component containing backup server j. For state (m, n, o, p, q), the number of unavailable functions is $(n+o)$. Hence, the average number of unavailable functions in the connected component is $\sum_{(m,n,o,p,q)\in U_j}(n+o)P(m, n, o, p, q)$, where $P(m, n, o, p, q)$ represents the probability that the system is in state (m, n, o, p, q). According to types 15 and 16, a function goes to the failed state with the transfer rate of $m\lambda$. According to type 17, a backup server goes to failed state and p functions go to failed state with the transfer rate of $\lambda(\omega)$. Therefore, $\sum_{(m,n,o,p,q)\in U_j}(m\lambda + pq\lambda(\omega))P(m, n, o, p, q)$ functions become unavailable per unit time. The Little's formula states that the time average number of units in system equals the arrival rate of units multiple the average time-in-system per unit [?]; by using the Little's formula, the average unavailable time of function $i \in L_j$ is computed by,

$$T_i^{\mathrm{UT}} = \frac{\sum_{(m,n,o,p,q)\in U_j}(n+o)P(m,n,o,p,q)}{\sum_{(m,n,o,p,q)\in U_j}(m\lambda + pq\lambda(\omega))P(m,n,o,p,q)} \quad (2)$$

Similarly, the average available time of function $i \in L_j$ is computed by,

$$T_i^{\mathrm{AT}} = \frac{\sum_{(m,n,o,p,q)\in U_j}(m+p)P(m,n,o,p,q)}{\sum_{(m,n,o,p,q)\in U_j}[(n+o)\mu + o\delta_j]P(m,n,o,p,q)} \quad (3)$$

Substituting (2), (3) into (1), we can get the unavailability Q_i of function $i \in L_j$.

IV. NUMERICAL RESULTS

We use the model with the constant failure probability of backup server as our baseline model. Then we compare the maximum unavailability of the proposed model to the baseline model.

As an experimental conditions, the entire system consists of 100 middleboxes and 20 backup servers, i.e., $|F| = 100, |S| = 20$. We randomly set the protection capacity c_j and the recovery capacity r_j for each backup server j from the range of $[1, M]$, and it has to satisfy a condition that $c_j \geq r_j$. The average recovery rate δ_j of

TABLE II
feasible system state transition incoming to (m, n, o, p, q)

Type	State	Transfer rate	conditions
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	$(m-1, n, o, p+1, q)$	$(p+1)\mu$	$m \geq 1$, $o+1+p \leq r_j$, and $q=1$
4	$(m-1, n, o-1, p+1, q)$	$(p+1)\mu$	$m \geq 1$, $o \geq 1$, and $o+p=r_j$
5	$(m, n+o, 0, 0, q-1)$	μ	$p=0$ and $q=0$
6	$(m+1, n-1, o, p, q)$	$(m+1)\lambda$	$n \geq 1$
7	$(m+1, n, o-1, p, q)$	$(m+1)\lambda$	$n=0$ and $o \geq 1$
8	$(m, n-o'-p', o', p', q+1)$	$\lambda(\omega)$	$q=0$
9	$(m, n, o+1, p-1, q)$	$(o+1)\delta_j$	$p \geq 1$

TABLE III
feasible system state transition outgoing from (m, n, o, p, q)

Type	State	Transfer rate	conditions
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	$(m, n-l, l, 0, q+1)$	μ	$q=0$
15	$(m-1, n+l, o, p, q)$	$m\lambda$	$m \geq 1$ and $o+p=r_j$
			$m \geq 1$ and $q=0$
16	$(m-1, n, o+1, p, q)$	$m\lambda$	$m \geq 1$, $o+1+p \leq r_j$, and $q=1$
17	$(m, n+o+p, 0, 0, q-1)$	$\lambda(\omega)$	$q=1$
18	$(m, n, o-1, p+1, q)$	$o\delta_j$	$o \geq 1$

each backup server j is randomly set within the range of $[L, U]$.

We use a simulated-annealing (SA) algorithm [?], [?] to find the optimal assignment that minimizes the maximum unavailability. We set a large initial temperature T_{init}, a low termination temperature T_{term} and a rate of temperature descent ρ; in each iteration, the temperature drops to the previous temperature times ρ, and the algorithm stops when the temperature is less than or equal to the termination temperature. In each iteration, we randomly select two backup servers, assign the protected function of one server to another backup server with idle protection capability, and compute the new maximum unavailability of the current assignment. If the current assignment has a lower maximum unavailability than before the change, accept it; otherwise, we accept the current assignment with a certain probability, which is to escape the local optimal assignment. In our experiment, the condition of SA algorithm is set as $T_{\text{init}} = 10000000$, $T_{\text{term}} = 0.00001$, and $\rho = 0.99$.

In order to simplify the experiment, we only analyze the case where $|H| = 1$. In the proposed model, the failure rate of backup server $\lambda(\omega)$ is a function that reflects the relationship between failure rate λ_j and workload ω. The workload, $\omega \in [0, 1]$, of backup server j is the ratio of the number of protected functions that are being recovered or have been recovered by the backup server to the recovery capability:

$$\omega = \frac{o+p}{r_j}. \tag{4}$$

We use a simple direct proportional function to describe the $\lambda(\omega)$:

$$\lambda(\omega) = 0.000001 + \omega * (\lambda_h - 0.000001). \tag{5}$$

If a constant failure rate is used to analyze the unavailability, we need to conservatively adopt the failure rate of server in the worst case. So the maximum of $\lambda(\omega)$ is equal to the constant failure rate of λ_h in the baseline model. When the workload is 0, that is, no function fails, the backup server has the lowest failure rate; we set it to a relatively low value of 0.000001. When the workload is 1, that is, all recovery capabilities of the backup server are occupied, the backup server has the highest failure rate, λ_h, which is the same as the constant failure rate of the baseline model.

In every trial, we first run the baseline model to get the assignment S_{st} and its maximum unavailability Q_{st} in the case of constant failure rate of backup server. Then we run the proposed model to obtain the assignment S_{wl} and its maximum unavailability Q_{wl} in the case of workload-dependent failure rate. Finally, we recompute the maximum unavailability Q_{stwl} of S_{st} in the case of workload-dependent failure rate and compare it with Q_{wl}.

For each set of settings, we run 500 trials and get the average maximum unavailability. Figure 3 shows the

Fig. 3. Comparison of maximum unavailability obtained by the proposed and baseline models with the different upper bounds, M

Fig. 4. Comparison of maximum unavailability obtained by the proposed and baseline models with the different failure rates, λ_h

maximum unavailability obtained by the proposed and baseline models with the different upper bounds, M, of c_j and r_j. Repair rate μ of functions and servers is set to 0.001; the failure rate of the function and the backup server in the baseline model is set to 0.0001; the average recovery time δ_j^{-1} is randomly set from the range of [10, 100]. It is observed that the obtained maximum unavailabilities of both models decrease with the increase of the upper bound, M, of c_j and r_j. This is because the larger recovery capacity r_j makes the failed functions in the system more likely to be in the being recovered and recovered state rather than the waiting state. In addition, larger protection capacity c_j increases the number of possible assignments, thereby increasing the chances of a potentially better assignment. The proposed model reduces the average maximum unavailability by an average of 17% compared to the baseline model.

Figure 4 shows the maximum unavailability obtained by the proposed and baseline models with the different average failure rates, λ_h. Repair rate μ of functions and servers is set to 0.001; the protection capacity c_j and the recovery capacity r_j are randomly set from the range of [1, 16]; the average recovery time δ_j^{-1} is randomly set from the range of [10, 100]. We find that the maximum unavailability increases as the average failure rate increases for both models. Since the larger failure rate makes functions in the system more likely to leave the active state, and be processed by the backup server or wait for the backup server. In addition, backup server j has a higher failure rate due to higher workload, which increases the probability that all the failed functions protected by j enter the waiting state due to its own failure. The proposed model reduces the average maximum unavailability by an average of 18% compared to the baseline model.

Fig. 5. Comparison of maximum unavailability obtained by the proposed and baseline models with the different repair rates, μ

Fig. 6. Comparison of maximum unavailability obtained by the proposed and baseline models with the different upper bounds, U

Figure 5 shows the maximum unavailability obtained by the proposed and baseline models with the different repair rates, μ. Failure rate λ_h of the function and the backup server in the baseline model is set to 0.0001; the protection capacity c_j and the recovery capacity r_j are randomly set from the range of [1, 16]; the average recovery time δ_j^{-1} is randomly set from the range of [10, 100]. It is observed that for both models, the obtained maximum unavailabilities decrease with the increase of repair rate μ_h. This is because the higher recovery rate allows failed functions to return to the active state more quickly, which increases the probability that functions are in the active state and reduces the workload on the backup server. The proposed model reduces the average maximum unavailability by an average of 7% compared to the baseline model.

Figure 6 shows the maximum unavailability obtained by the proposed and baseline models with the different upper bounds, U, of the average recovery time, δ_j^{-1}. The lower bound L of δ_j^{-1} is set to 10; the protection capacity c_j and the recovery capacity r_j are randomly set from the range of [1, 16]; the repair rate μ of functions and servers is set to 0.001; the failure rate of the function and the backup server in the baseline model is set to 0.0001. It is observed that for both models, the obtained maximum unavailabilities increase with the increase of L. This is because larger U makes the failed function spend more time in the being recovered state. The proposed model reduces the average maximum unavailability by an average of 19% compared to the baseline model.

V. CONCLUSION

This paper proposed a shared backup allocation model of middlebox with considering the failure, repair, recovery rate of each element and the workload of backup server. Multiple functions can share the backup resources on a backup server. By analyzing the process of failure, repair and recovery, we can get four possible states for each function, and the system state transition. Through the queuing approach, we can compute the probability of each function being available or unavailable in a certain assignment, and obtain the unavailability of each function. The proposed model aims to find an assignment to minimize the maximum unavailability among functions. We solve this problem by the simulated annealing algorithm. We evaluate and compare performance of proposed and baseline models under different experimental conditions. Based on the results, we can observe that, compared to the baseline model, the proposed model reduces the average maximum unavailability by an average of 15% in our examined cases.