Hybrid Recovery-Based Intrusion Tolerant System for Practical Cyber-Defense

Bumsoon JANG\textsuperscript{(a)}, Student Member, Seokjoo DOO\textsuperscript{††}, Soojin LEE\textsuperscript{†††}, and Hyunsoo YOON\textsuperscript{†}, Nonmembers

SUMMARY Due to the periodic recovery of virtual machines regardless of whether malicious intrusions exist, proactive recovery-based Intrusion Tolerant Systems (ITSs) are being considered for mission-critical applications. However, the virtual replicas can easily be exposed to attacks during their working period, and additionally, proactive recovery-based ITSs are ineffective in eliminating the vulnerability of exposure time, which is closely related to service availability. To address these problems, we propose a novel hybrid recovery-based ITS in this paper. The proposed method utilizes availability-driven recovery and dynamic cluster resizing. The availability-driven recovery method operates the recovery process by both proactive and reactive ways for the system to gain shorter exposure times and higher success rates. The dynamic cluster resizing method reduces the overhead of the system that occurs from dynamic workload fluctuations. The performance of the proposed ITS with various synthetic and real workloads using CloudSim showed that it guarantees higher availability and reliability of the system, even under malicious intrusions such as DDoS attacks.

key words: intrusion tolerant system (ITS), hybrid recovery, availability-driven recovery, dynamic cluster resizing, mission-critical application

1. Introduction

With the rapid growth of communication and network technologies, information systems have provided various online mission-critical services in areas such as the military, healthcare and power grids. Since the web services for these critical applications must be available in 24/7, availability, reliability and security are vital factors for the information systems [1], [2].

Several security solutions, such as Intrusion Detection Systems (IDSs), Intrusion Protection Systems (IPSs), and Firewalls, have been widely used to supply stable services without any performance degradation of the systems, even in the presence of intrusions or errors. In particular, intrusion tolerance has been a good approach to achieve the practical goal of cyber-defense that has to provide services even if crucial parts of the systems are taken by attackers [3].

Proactive recovery-based Intrusion Tolerant Systems (ITSs) using virtualization techniques have been the most actively studied and have shown that they are suitable models for arbitrary faults, called Byzantine faults [4]. In proactive recovery-based ITSs, virtual replicas as service components are recovered periodically to eliminate intentional intrusions or potential errors. However, proactive recovery-based ITSs can be attacked and made to fail by intelligent attackers if the attackers can obtain privileges to take over the system within the working period of the virtual replicas. To deal with this weakness, it is necessary for proactive recovery to adopt a complementary procedure, namely reactive recovery for mission-critical services.

In this paper, we propose a novel hybrid (proactive and reactive) recovery-based intrusion tolerant architecture. We use an availability-driven recovery scheduling method for the hybrid recovery to lower the exposure time for higher availability of the system and to prevent the performance degradation caused by stealthy intrusions like application-layer DDoS attacks [5]. Moreover, we present dynamic cluster resizing to provide consistent services against dynamic workload fluctuations in real-time system environments. Thus, it makes the system more resistant to volumetric DDoS attacks that disrupt the application’s operation.

The proposed architecture was simulated by CloudSim and the experimental results proved that the proposed model is effective in terms of availability and reliability for ITSs. The rest of the paper is organized as follows. In Sect. 2, related works based on recovery-based ITSs are surveyed. Section 3 presents the proposed hybrid recovery-based ITS architecture. Then, the experimental results and evaluation of the proposed architecture are shown in Sect. 4. Finally, Sect. 5 concludes the paper.

2. Related Work

Intrusion tolerant began with Byzantine Fault Tolerance (BFT) replication to maintain the integrity and availability of the system from arbitrary faults [6]–[8]. A BFT system is able to provide system services with correctly functioning components even if n is greater than fault components f. Proactive recovery with BFT replication [9], [10] is a good approach for ITSs. Since proactive recovery executes recovery of components periodically whether they are faulty or not, the malicious adversary can only stay during a component’s working duration, which is called the exposure time.

Previous researches have shown that proactive recovery-based ITSs are cost-effective, easily controlled and
elastically applicable due to the benefits of virtualization techniques [11]–[15]. Self Cleansing Intrusion Tolerance (SCIT) is a typical example of proactive recovery-based ITSs based on the virtualization technique [16], [17]. In SCIT, each Virtual Machine (VM) is refreshed periodically in serial order with a fixed exposure time, which is the working time of the VM for providing services. However, SCIT does not consider the volumetric DDoS or the flash crowd problem [18], which is a sudden increase of service requests caused by dynamic workload shifting. Lim et al. [19] proposed the Adaptive Cluster Transformation (ACT) with proactive recovery-based ITSs. They enhanced the Quality of Service (QoS) under the flash crowd problem or the volumetric DDoS. Nevertheless, this approach did not address appropriate exposure time management while the cluster size was altered. It also did not have any protection mechanism against stealthy attacks like application-layer DDoS attacks.

Sousa et al. [20] showed that proactive recovery-based ITSs can be vulnerable if malicious adversaries are smart enough to intrude into many virtual replicas simultaneously during the service component’s life time. To handle this defect, several researchers [21], [22] used an additional process called reactive recovery. In the Adaptive Recovery Scheme (ARS) [23], the faulty virtual replicas are rejuvenated by the proactive-reactive recovery and their exposure time is regulated adaptively to guarantee the system’s availability and performance. However, this approach could not handle traffic surges that can cause service faults or resource exhaustion attacks that may lead system failures.

Our study is founded on some of the previous studies with innovative methods for seamless services as the goal of practical cyber-defense. Compared with the other studies, our method better satisfied availability and reliability in both synthetic and real server workloads even in the presence of attacks. Moreover, the proposed architecture can be used for small-scale and large-scale services because of dynamic scalability by virtualization techniques.

3. Design of a Hybrid Recovery-Based Intrusion Tolerant System

In order to provide better availability and reliability than previous studies in real server workloads even in the presence of attacks, we designed the hybrid recovery-based ITS illustrated in Fig. 1.

In the hosting environment for mission-critical web services, clients are able to access services on the virtual replicas in a cluster, which is a set of replicas for the rejuvenation cycle, through a proxy. This architecture contains two main modules, particularly the hybrid recovery actuator using availability-driven recovery scheduling and the elastic cluster manager for the dynamic cluster resizing. There is also an ancillary module called the system health monitor, which observes information associated with the state of the virtual replicas.

For the proposed architecture, the following assumptions are applied.

- Service requests are time specific for mission-critical applications [24], [25].
- The modules are separated from external networks [16].
- All the known attacks are filtered out by existing security mechanisms such as firewalls, IPSs or IDSs.
- Each virtual replica has a different operating system and an application for diversity.
- Service requests are distributed to virtual replicas by a load balancer of the proxy server.

3.1 Availability-Driven Recovery Scheduling

The hybrid recovery actuator operates proactive and reactive recovery based on the availability-driven recovery scheduling to maintain the minimum exposure time as well as to avoid the damage by stealthy resource exhaustion attacks. Based on this scheduling method, we managed to improve the system availability over that of existing architectures.

3.1.1 Proactive Recovery with Recovery Ratio

Availability, which is the ability of a system to continue the service without any disturbance, is one of the important attributes for the system’s QoS. From previous proactive recovery-based ITSs studies, the availability of a system can be calculated with the following equation [26].

\[
\text{Availability} = \frac{h_G + h_V + P_a * h_A}{h_G + h_V + P_a * h_A + P_c * (1-P_c) * h_A}
\]

where \( P_a \) and \( P_c \) mean the probability of state transition from vulnerable to attacked and from attacked to good, respectively. \( h_G \), \( h_V \), and \( h_A \) are mean sojourn time in each state: (good, vulnerable, attacked).

The availability decreases monotonically with \( P_a \) but increases with \( P_c \). That is, the availability has an inverse relationship to the exposure time. This points out that minimizing exposure time will maximize the system’s availability. In this respect, an appropriate recovery scheduling method could be implemented to restrict the exposure time.
It should be noted that previous studies in recovery-based ITSs using cluster resizing did not take into account proactive recovery scheduling to minimize the exposure time.

For proactive recovery that eliminates the potential threat of stealth (or unknown) intrusions or restricts the residence time of malicious adversaries repeatedly, virtual replicas are in one of four states on the rejuvenation cycle [26]: ACTIVE - GRACE PERIOD - INACTIVE - LIVE SPARE.

- **ACTIVE**: A virtual replica is online to accept and process requests.
- **GRACE PERIOD**: A virtual replica is online and does not accept any new requests. It only processes pending requests.
- **INACTIVE**: A virtual replica is offline and is being recovered to pristine image.
- **LIVE SPARE**: A virtual replica is offline and waits for the next online services.

This procedure is divided into two phases: working and cleansing phases. During the working phase, which includes ACTIVE and GRACE PERIOD, virtual replicas are online and provide services. During the cleansing phase, which includes INACTIVE and LIVE SPARE, the replicas are offline. The exposure time of a virtual replica is the time for the working phase.

Based on this concept, the cleansing time $T_c$, the exposure time $T_e$, the number of working replicas $N_w$, and the number of cleansing replicas $N_c$ satisfy the following equation [23]:

$$T_e = \frac{T_c \ast N_w}{N_c}$$  \hspace{1cm} (2)

To maximize the resource utilization, virtual replicas should be cleansed in order. In other words, the number of cleansing replicas $N_c$ is set to 1. Figure 2 shows the rejuvenation order model, which has three working replicas with one cleaning replica and seven working replicas with one cleansing replica. It was found that the exposure time $T_e$ increased as the size of a cluster increased [19], and it resulted in a lower availability of the system.

Thus, we can conclude that the same recovery ratio $R \propto \frac{N_w}{N_c}$ should be preserved in any circumstances to prevent the exposure time $T_e$ from increasing by an extended cluster size. The following equation shows the relation between $T_e$ and $R$.

$$T_e = T_c \ast R$$  \hspace{1cm} (3)

Because the cleansing time $T_c$ is not adjustable [13], [21], the exposure time depends on only the recovery ratio $R$.

Considering the Byzantine Fault, the total number of replicas $N_t$ can be defined as

$$N_t = N_w + N_c = (2 \ast f + 1) + r$$  \hspace{1cm} (4)

where the values of $f$ and $r$ represent the number of intrusion and concurrent recovering replicas, respectively. When a system operator chooses the recovery ratio $R$ and the number of working replicas by intrusion $f$ is settled, the number of recovering replicas $N_c$ should meet the following equation:

$$R \propto \left[ \frac{N_w}{N_c} \right] = \left[ \frac{2 \ast f + 1}{N_c} \right]$$  \hspace{1cm} (5)

This implies that, during the cleansing phase, $N_c$ replicas should be supplied simultaneously for proactive recovery to prevent increasing the exposure time so that the system can maintain a higher availability with a small amount of exposure time.

### 3.1.2 Reactive Recovery with CPU Utilization

Recent studies [20], [23], [27] found that proactive recovery had an inherent limitation by periodical recovery. Even if proactive recovery can eliminate some unknown attacks like zero-day attacks periodically [28], stealth resource exhaustion attacks like an application-layer DDoS directly causes the performance degradation, and they cannot be tolerated easily by existing proactive recovery or reactive recovery which use the signature based intrusion detection. However, since such attacks could interrupt services seriously or damage the performance of the system by exhausting system resources [29], they can be noticed easily by the resource monitoring.

In this paper, the CPU utilization $U$ can be computed as the ratio of the total running time of requests to the unit time:

$$U = \frac{\sum_{i=1}^{n} t_i}{t_{unit}}$$  \hspace{1cm} (6)

where $t_i$ and $t_{unit}$ denote the processing time of the $i$-th request and the unit time respectively.

If there exists a virtual replica with almost full CPU utilization ($U \approx 100\%$) in the ACTIVE state, then the state of the replica changes into the GRACE PERIOD state. Nevertheless, if the replica in the GRACE PERIOD state is still in

| Cluster Size=3 | VM1 | Online | Offline | Online | Online | Online | Online | -- |
|----------------|-----|---------|---------|--------|--------|--------|--------|----|
| Cluster Size=7 | VM1 | Online | Offline | Online | Online | Online | Online | Online |
|                | VM2 | Online | Offline | Online | Online | Online | Online | Online |
|                | VM3 | Online | Offline | Online | Offline | Online | Online | --   |
|                | VM4 | Offline | Online | Online | Online | Online | Online | --   |
|                | VM5 | Online | Online | Online | Online | Offline | Online | --   |
|                | VM6 | Online | Online | Online | Offline | Online | Online | --   |
|                | VM7 | Online | Online | Online | Online | Offline | Online | --   |
|                | VM8 | Offline | Online | Online | Online | Online | Online | --   |

**Fig. 2** Cleansing order model with the cluster size of 3 and 7
full CPU utilization, the processing request can be considered as a suspicious request to exhaust the system resources and it will be canceled immediately for the system capability of maintaining QoS. The unprocessed requests in the virtual replica to be instant recovery will be distributed to other virtual replicas in the ACTIVE state. However, if the CPU of the virtual replica in the GRACE PERIOD state is not fully utilized, a normal recovery process is applied to recover the virtual replica.

The availability-driven recovery scheduling algorithm consisting of a proactive recovery with a controlled recovery ratio and a reactive recovery with CPU monitoring is explained in Algorithm 1.

3.2 Dynamic Cluster Resizing

The dynamic cluster resizing mechanism aims at continuous and trustworthy services of the system even when there is dynamic workload variation or a volumetric DDoS. There are three processes in the elastic cluster management for the dynamic cluster resizing scheme: cluster scale up, cluster scale down, and double-plus-one expansion.

The dynamic workload fluctuation, including a flash crowd problem, could be moderated by the cluster scale up and down processes. A severe case, like service failure caused by an extremely high load of requests or stealthy resource exhaustion attacks, are mitigated by the double-plus-one expansion. Even when the high-load requests are not malicious but the volume of requests is huge, the deadly effect of the high-load requests on the service provision should be managed by the double-plus-one expansion as in case of the volumetric DDoS for seamless service. Consequently, if high-load requests trigger the double-plus-one expansion due to a disruption of service, it should be considered as a volumetric DDoS attack.

An appropriate dynamic cluster resizing method is chosen by the manager based on CPU utilization, queue size, and success rate. Note that, in ACT[19], only the response time is used for the adaptive cluster transformation. However, more than one factor, such as CPU utilization, queue size, and success rate, can be used to help the control reduce the overhead by a cluster size change.

The CPU utilization $U$ can be obtained from Eq. (6). The queue size $Q$ indicates the number of unprocessed waiting requests in each virtual replica. This value is used to cope with rapidly fluctuating workloads.

Generally deadline-constrained service delivery is required in real-time systems [30]. Even though the service result is correct, the service is regarded as a failure if its delay, called response time, is not acceptable according to time limits for delay-sensitive applications. For this reason, the success rate as a deadline constraint is an efficient factor to see how good the system operates in real-time [25]. The success rate $S$ can be derived by:

$$ S = \frac{RQ_{\text{limit}}}{RQ_{\text{total}}} $$

where $RQ_{\text{limit}}$ is the number of requests responded to within the time limit and $RQ_{\text{total}}$ is the total number of requests.

Monitoring is needed to measure three factors: the CPU utilization ($U$), the queue size ($Q$), and the success rate ($S$). These factors are used for the proper operation of the hybrid recovery actuator and the elastic cluster manager. Thus, accurate and fine-grained monitoring of the system health monitor is essential.

3.2.1 Cluster Scale UP & DOWN

Increasing the cluster size is a critical process for maintaining higher system performance because more virtual replicas as service components are needed for substantial requests. When the average CPU utilization of virtual replicas in the working phase is more than 80% and the success rate per unit time is less than 90%, the dynamic cluster manager provides two more virtual replicas in the cluster to maintain the recovery ratio.

On the other hand, there is always a cost issue with resources in the virtualization environment. In this respect, decreasing the cluster size is desirable for a small number of service requests. However, the retrieval of virtual replicas from a cluster needs to be more precise because it could ag-

---

**Algorithm 1 Availability-Driven Recovery**

**Require:**

$T_{cur}$: current time

$T_{pre}$: previous time in cleansing period

$T_{die}$: cleansing period time

$AT$: the list of VM in ACTIVE state

$GP$: the list of VM in GRACE PERIOD state

$IA$: the list of VM in INACTIVE state

$VM[i]$: $i$th VM in working phase (ACTIVE and GRACE PERIOD)

$Q[i]$: the CPU utilization of $i$th VM

$Q[i]$ = unprocessed request in queue of $i$th VM

1: **Proactive Recovery:**

2: if $T_{cur} - T_{pre} = T_{die}$ then

3: if $(\text{size}(AT[i]) + \text{size}(GP[i])) \mod R = 0$ then

4: $N_s = (\text{size}(AT[i]) + \text{size}(GP[i]))/R$

5: for $i = 0$ to $N_s$ do

6: POP $VM[0]$ from $AT[i]$

7: PUSH $VM[0]$ into $GP[i]$

8: end for

9: else

10: $N_s = ((\text{size}(AT[i]) + \text{size}(GP[i]))/R) + 1$

11: for $i = 0$ to $N_s$ do

12: POP $VM[0]$ from $AT[i]$

13: PUSH $VM[0]$ into $GP[i]$

14: end for

15: end if

16: $T_{pre} = T_{cur}$

17: end if

18: **Reactive Recovery:**

19: for $i = 1$ to $N_s$ do

20: if $U[i] \geq 100\%$ and $VM[i]$ in $AT[i]$ then

21: POP $VM[i]$ from $AT[i]$

22: PUSH $VM[i]$ into $GP[i]$

23: else if $U[i] \geq 100\%$ and $VM[i]$ in $GP[i]$ then

24: POP $VM[i]$ from $GP[i]$

25: for all $Q[i]$ do

26: ASSIG $Q[i]$ to $AT[i]$

27: end for

28: PUSH $VM[i]$ into $IA[i]$

29: end if

30: end for
gravitate the system performance significantly in a fallacious case. This could lead to frequent reconfiguration of the cluster size. When the average CPU utilization is less than 50% and the overall success rate is over 80%, the system asks for an increment of resource utilization by reducing two surplus virtual replicas.

The threshold values for cluster scale up and down control are set arbitrarily considering the proper CPU utilization level for the seamless service and acceptable degree of the deadline-constrained service. In practice, these values can be determined using the trial-and-error method [31].

3.2.2 Cluster Double-Plus-One Expansion

Even if the system manages incoming request variation well by increasing and decreasing cluster processes, the extreme cases like service failures by volumetric DDoS attacks or Byzantine fault by application-layer DDoS attacks should also be considered in the proposed architecture. In ACT [19], the volumetric DDoS was mitigated by a cluster substitution that replaced the existing cluster with a new one. However, since the overhead of the substitution process is huge, the size of the cluster is not changeable during DDoS attacks. Moreover, ACT did not consider application-layer DDoS attacks.

For flexible cluster resizing to endure massive incoming requests and to mitigate both the volumetric DDoS and application-layer DDoS attacks in advance, the elastic cluster manager chooses cluster expansion using a double-plus-one strategy ($n = 2n + 1$). Since the system with new cluster size $n'$ has more service components than the previous state, the proposed architecture will not suffer from the volumetric DDoS attacks or the service failures by Byzantine fault ($n' > 2f + 1$).

When the number of unprocessed requests in the queue is greater than the result of the system capacity $C$ multiplied by $m$, which is determined by the operator, or a Byzantine fault occurring from the application-layer DDoS attacks ($f > \frac{m-1}{2}$), the double-plus-one expansion is activated. An unavailable VM from application-layer DDoS attacks can be managed by instant cleansing, but there is a possibility of a Byzantine fault by multiple simultaneous attacks. Therefore, to defeat a Byzantine fault by application-layer DDoS attacks, double-plus-one expansion is performed when the number of virtual replicas with full CPU utilization in the working phase is greater than $\frac{(n+1)^2}{2}$.

The above-mentioned elastic cluster manager module for dynamic cluster resizing is shown as pseudo code in Algorithm 2. The threshold values of CPU utilization $U$ and success rate $S$, and multiplier $m$ can be adjusted by the operator for the system environment or the service requirement.

4. Experimental Evaluations

We performed a series of experiments to prove the high availability and reliability with low overhead of the proposed architecture in dynamic workload environments. We compared the performance of the proposed architecture with the results of ACT [19] and ACT with the recovery ratio (ACT-RR), which is a modified ACT using the response time, the longest exposure time, and the number of times the cluster size changed in each experiment. Since SCIT [16] and ARS [23] are recovery-based ITSs using a fixed cluster size, they were excluded from the comparison targets to ensure impartial evaluation.

In this study, we performed experiments in the CloudSim simulator [32]. The CloudSim is a special simul-

---

**Algorithm 2 Dynamic Cluster Resizing**

**Require:**
- $T_{curr}$: current time
- $T_{att}$: adjustment time for the cluster resizing effect
- $T_{dec}$: decision time of the cluster resizing
- $N_r$: the number of working VMs
- $N_{max}$: the maximum number of working period VMs for the resource limitation
- $N_{min}$: the minimum number of working period VMs for the service guarantee
- $AT[]$: the list of VM in ACTIVE state
- $GP[]$: the list of VM in GRACE PERIOD state
- $IA[]$: the list of VM in INACTIVE state
- $LS[]$: the list of VM in LIVE SPARE state
- $V M[]$: $i$th VM in working phase (ACTIVE and GRACE PERIOD)
- $U$: the CPU utilization of $i$th VM
- $U_{avg}$: the average of CPU utilization during unit time
- $S$: the Success Rate of requests during unit time
- $S_{ext}$: the Success Rate of requests so far
- $Q_{avg}$: the average of unprocessed request in queue during unit time
- $C$: the capacity of system (the system throughput per unit time)
- $m$: the multiplier for volumetric DDoS prediction

1: if $T_{curr} \neq 0$ and $T_{curr} < T_{att}$ then return 0
2: end if
3: if $T_{curr} \neq 0$ and $T_{curr} - T_{att} < T_{dec}$ then
4: SCALE UP:
5: if $mean(U[i]) >= 0.8$ and $S <= 0.9$ and $(size(AT[i]) + size(GP[i])) + 2 <= N_{max}$ then
6: $N_r += 2$
7: count = 0
8: while count < 2 do
9: $PO PM[i][0]$ from $AT[i]$ or $LS[i]$
10: $PUSH VM[0]$ into $AT[i]$
11: count ++
12: end while
13: $T_{curr} = T_{att}$
14: end if
15: end if
16: SCALE DOWN:
17: if $mean(U[i]) <= 0.5$ and $S_{ext} >= 0.8$ and $(size(AT[i]) + size(GP[i])) - 2 >= N_{min}$ then
18: $N_r -= 2$
19: count = 0
20: while count < 2 do
21: $PO PM[i][0]$ from $AT[i]$
22: $PUSH VM[0]$ into $GP[i]$
23: count ++
24: end while
25: $T_{curr} = T_{att}$
26: end if
27: if $AT[] (\geq size(V M[i]) / 2$ or $Q_{avg} > C \times m$) and $(size(AT[i]) + size(GP[i])) \times 2 + 1 <= N_{max}$ then
28: $N_r = N_r + 2 + 1$
29: count = 0
30: while count < ($N_r + 1$) do
31: $PO PM[i][0]$ from $AT[i]$ or $LS[i]$
32: $PUSH VM[0]$ into $AT[i]$
33: count ++
34: end while
35: $T_{curr} = T_{att}$
36: end if
lation toolkit for modeling and simulation for virtualization environments. The experiments were simulated in a Windows 7 (64bit) system with a 3.3GHz Intel i3 processor and 16GB memory. The proposed architecture and scenarios were written in Java language in Eclipse IDE environments. The detailed implementation environment was as follows:

- Each virtual replica had one processing element with a capacity of 512MIPS.
- The minimum number of required virtual replicas for granting service was 3 VMs \((2f + 1)\).
- The maximum number of operational virtual replicas in the hosts of the system was 20 VMs.
- The number of service requests was 1,000,000 in total for synthetic cases.
- The response deadline, which is the acceptable time limit for delay-sensitive applications, was 0.77 seconds (small instance of Amazon EC2) [33].
- The workload size for a request was a replica capacity \((512\text{MIPS} \times \text{response deadline})\).
- The recovery period (the duration time of INACTIVE state) was 146 seconds [21].
- The recovery ratio \((R, \text{the ratio between the number of working VMs and cleansing VMs})\) was 3.
- The application-layer DDoS followed a Poisson process with a mean value of 0.000004456 (frequency of the major DDoS attacks) [34].
- The volumetric DDoS was simulated as massive requests in a very short time (the two-pulse workload).

For the experimental design, we considered the proposed architecture as a system for the delay-sensitive application. Therefore, the workload size for a request was assumed as a replica capacity \(\times \text{response deadline}\) to meet the deadline constraint in a normal situation. Furthermore, the minimum number of required virtual replicas for granting the service was assumed as 3 VMs considering a Byzantine Fault \((2f + 1)\) and the number of cleansing virtual replicas was expected as 1 VMs to maximize the resource utilization. Therefore, the recovery ratio \(R\) was set to 3 using Eq. (5).

In the experiments, we used three synthetic workload types that represented different workload fluctuations. In addition the workload of EPA-HTTP [35] was used for a realistic scenario of web-based systems. Four types of workloads were as follows and are illustrated in Fig. 3.

- Normal: average request arrival rate
- Stair-step: gradual increment and similar decrement of request arrival rate
- Two-pulse: sharply repeated increments of request arrival rate (volumetric DDoS)
- EPA-HTTP: actual HTTP requests rate for a day from a busy server

4.1 Effects of the Recovery Ratio on Exposure Time

To investigate the effect of using the recovery ratio, the exposure time for a simple recovery using one cleansing replica was compared to one for the proactive recovery scheduling with a recovery ratio \(R = 3\). Figure 4 compares the measured exposure time for three different synthetic workload types. As mentioned in the previous section, when the size of the cluster increased from 3 to 15, the exposure time of the simple recovery using a single cleansing replica increased up to about 3300∼3500s, which was 5 times the lowest exposure time with the 3VMs cluster. However, the exposure time of the proactive recovery scheduling using \(R = 3\) was maintained at below about 550∼600s except for the 3VMs cluster case on a two-pulse workload (550∼1150s) as shown in Fig. 4 (c). The increased exposure time means that the system is more vulnerable because malicious attackers can take much more time to find the vulnerabilities of the system. If an attacker already successfully penetrated the system, the attacker’s residence time will be also increased for the next steps, like privilege escalation, maintaining and expanding access, etc.

As a result, the controlled recovery ratio can appropriately maintain the minimum exposure time, so that the availability and security of the system is guaranteed.

4.2 Performance Comparison of Intrusion Tolerance

Figures 5, 6, 7, and 8 describe the performance comparison among the proposed architecture (black-cross line), ACT (blue-circle line), and ACT-RR (red-triangle line) with the four different workloads: normal, star-step, two-pulse and EPA-HTTP. Table 1 gives the total number of times the cluster size changes.

In the experiment, the availability, reliability, and overhead of the proposed architecture was represented as the response time, the longest exposure time, the success rate, and the number of cluster size changes. As an index of attack risk, the longest exposure time of any virtual replica is a
meaningful value to represent the attacker’s residence time.

Figures 5 (a), (b), and (c) show that the proposed architecture’s response time, longest exposure time, and success rate were similar to those of ACT and ACT-RR in normal workload. The response time, the longest exposure time, and the success rate of the proposed architecture stabilized at around 0.76s, 300∼900s, and 80%, respectively. In particular, in Fig. 5 (c), the success rate of the proposed architecture was about 90% in the period of 250000∼400000s due to two times of the cluster scale up process. Moreover, with the application-layer DDoS in Figs. 5 (d), (e), and (f), the response time of the proposed architecture was slightly shorter than the others and the longest exposure time was less than 900s. The others’ longest exposure time went up linearly according to the elapse of time because the application-layer DDoS attack prevented the cleansing process by occupying the resources. Success rate of the proposed architecture showed over 70% while the others’ success rate was under 70%. Further, the number of cluster size changes, which indicates the frequency of VM creation and deletion as overhead for the cluster operation, was much lower than the others (about 90 times) under the application-DDoS attack in Table 1.

In a similar way, Figs. 6 and 7 show that the performance of the proposed architecture is much better than the other architectures, even with dynamic workload fluc-
tations (the stair-step workload) or volumetric DDoS (the two-pulse workload) as well as application-layer DDoS as stealthy resource exhaustion attacks. The response time and the success rate of the proposed architecture were maintained at below about 0.77s, which is the designated response deadline and above the higher success rate of 70%.

Fig. 6  Performance on stair-step workload

Fig. 7  Performance on two-pulse workload
In addition, the longest exposure time of the proposed architecture was maintained at under 900s, while the others weren’t. Even though there was a similar result of the longest exposure time in Fig. 6 (b), the number of cluster size changes in ACT and ACT-RR was much higher than the proposed architecture (about 90 times) as seen in Table 1. Moreover, in ACT-RR a system failure occurred from the application-layer DDoS attack at about 300000s, respectively (Figs. 6 (e) and 7 (e)). In CloudSim, a system failure was noticed by a user-defined function to count the returning requests that were processed through the VMs. Therefore, the system was considered as failed when a request sent to the VMs was working but the counting function was not working anymore.

To evaluate the performance of the proposed architecture in a realistic environment, an experiment using a real HTTP log, which is an EPA-HTTP one-day request log, was conducted. In Figs. 8 (a), (b), and (c), the response time and the success rate between the proposed architecture and the others are similar. However, the longest exposure time of ACT and ACT-RR was up to 1300s and 1000s, respectively, while that of the proposed architecture held at under 900s during a 50000–60000 elapsed time. Figures 8 (d), (e), and (f) show that the proposed architecture manages the circumstances of the security threat well for higher availability and reliability with the restricted longest exposure time. Similar to the other workloads, the total number of cluster size alterations with the EPA-HTTP workload was lower than those of the others as shown in Table 1.

Figures 5, 6, 7, and 8 confirm that the proposed architecture operates in the proactive way (ACT and ACR-RR) under a normal workload. Under threats, however, cluster resizing and the reactive mechanism were triggered to mitigate the heavy workload and the application-layer DDoS attack, while the proactive recovery using the recovery ratio prevented the exposure time increasing. Hence, the performance of the response time, the longest exposure time, the success rate and the number of cluster size changes were stabilized without any skyrocketing in the performance as shown in the above experimental result.

Thus, the proposed architecture using availability-driven recovery and dynamic cluster resizing managed the exposure time effectively to maximize the service availability and maintained a certain level of appropriate response time, even with workload fluctuations. Moreover, this ar-

### Table 1: The number of times that the cluster size changes

| Workload types | Architecture | Application-layer DDoS without | Application-layer DDoS with |
|----------------|--------------|--------------------------------|-----------------------------|
| Normal         | ACT          | 0                              | 906                         |
|                | ACT-RR       | 0                              | 906                         |
|                | Proposed Arch. | 2                             | 10                          |
| Stair-step     | ACT          | 556                            | 397                         |
|                | ACT-RR       | 560                            | 956 (Failure)              |
|                | Proposed Arch. | 6                             | 22                          |
| Two-pulse      | ACT          | 301                            | 647                         |
|                | ACT-RR       | 296                            | 688 (Failure)              |
|                | Proposed Arch. | 17                            | 15                          |
| EPA-HTTP       | ACT          | 10                             | 12                          |
|                | ACT-RR       | 10                             | 98                          |
|                | Proposed Arch. | 8                             | 10                          |
architecture also mitigated the application-layer DDoS and volumetric DDoS attacks. It should be taken into account that ACT was introduced as a cluster resizing scheme using an ideal response time as the threshold value, but the ideal response time has to be measured in every cluster size in advance [19]. However, measuring the ideal response time for each cluster size is not practical, and in our experiments there were no differences among the ideal response times of more than 9 VMs clusters. Since the proposed approach does not need any pre-experiments to determine the empirical threshold value, it will be practical for applications to systems at various scales. In addition, using more than one threshold factors for more subtle cluster resizing makes it possible to restrict unnecessary cluster size changes.

5. Conclusion

This paper presented a novel hybrid recovery-based ITS architecture that uses proactive and reactive recovery to guarantee seamless services in mission-critical applications that require service availability and reliability for practical cyber-defense. For this purpose, availability-driven recovery scheduling consists of proactive recovery with a controlled recovery ratio to maintain an appropriate exposure time and reactive recovery with CPU monitoring to respond immediately to stealthy resource exhaustion attacks. We also adopted dynamic cluster resizing using the system’s internal factors for consistent service and lower overhead under dynamic workload fluctuations.

Various simulations were done with synthetic and real workloads, which also demonstrated that the proposed architecture operated effectively with higher availability while maintaining an appropriately short exposure time and desirable success rate in any type of workload. Furthermore, the results showed that the proposed architecture provides reliable services with a low overhead of cluster scaling, even in dangerous situations, such as DDoS attacks.

Based on this study, we were able to overcome the limitations of previous research on the recovery-based ITSs and to combine a hybrid recovery-based ITS with mission-critical applications to meet the goal of practical cyber-defense. Clearly, the proposed hybrid recovery-based intrusion tolerant architecture has promise as a tool in the cyber-defense of many mission-critical applications.

Acknowledgments

This research was supported by the National Research Foundation of Korea grant (NRF-2014R1A2A2A01006957) and Institute for Information & communication Technology Promotion (IITP) grant funded by the Korea government (MSIP) (10041244, R-20150223-000167).

References

[1] W. Zhao, “Towards practical intrusion tolerant systems,” Information and Communications Technologies (IETICT 2013), IET International Conference on, pp.280–287, 2013.
[2] M. Tanha and F. Hashim, Towards a secure and available smart grid using intrusion tolerance, vol.7646, pp.188–201, Springer, 2012.
[3] M. Atighechi, P. Pal, F. Webber, R. Schantz, C. Jones, and J. Loyall, “Adaptive cybersecurity for survival and intrusion tolerance,” Internet Computing, IEEE, vol.8, no.6, pp.25–33, 2004.
[4] C. Cachin and J.A. Poritz, “Secure intrusion-tolerant replication on the internet,” Dependable Systems and Networks, 2002, DSN 2002, Proceedings, International Conference on, pp.167–176, IEEE, 2002.
[5] M. Ficco and M. Rak, Intrusion tolerance of stealth dos attacks to web services, vol.376, pp.579–584, Springer, 2012.
[6] A.N. Bessani, “From byzantine fault tolerance to intrusion tolerance (a position paper),” Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on, pp.15–18, IEEE, 2011.
[7] M. Correia, N.F. Neves, and P. Verissimo, “Bft-to: Intrusion tolerance with less replicas,” The Computer Journal, p.bxs148, vol.56, no.6, pp.693–715, 2013.
[8] G.S. Veroneze, M. Correia, A.N. Bessani, L.C. Lung, and P. Verissimo, “Efficient byzantine fault-tolerance,” Computers, IEEE Transactions on, vol.62, no.1, pp.16–30, 2013.
[9] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” ACM Transactions on Computer Systems (TOCS), vol.20, no.4, pp.398–461, 2002.
[10] F. Wang, R. Uppalli, and C. Killian, “Analysis of techniques for building intrusion tolerant server systems,” Military Communications Conference, 2002, MILCOM’02, 2002 IEEE, pp.729–734, IEEE, 2003.
[11] M. Guo and P. Bhattacharya, “Diverse virtual replicas for improving intrusion tolerance in cloud,” Proceedings of the 9th Annual Cyber and Information Security Research Conference (2014), pp.41–44, ACM, 2014.
[12] G. Tian and D. Meng, “Implications of recovery schemes for virtualization platform,” Parallel Processing Workshops (ICPPW), 2011 40th International Conference on, pp.176–183, IEEE, 2011.
[13] F. Zhao, M. Li, W. Qiang, H. Jin, D. Zou, and Q. Zhang, “Proactive recovery approach for intrusion tolerance with dynamic configuration of physical and virtual replicas,” Security and Communication Networks, vol.5, no.10, pp.1169–1180, 2012.
[14] J.-H. Huang, J. Huang, R. Li, and X.-M. Li, “Virtualization-based recovery approach for intrusion tolerance,” Inform. Technol. J, vol.12, no.3, pp.385–390, 2013.
[15] R. Sindoori, V.P. Pallavi, and P. Abinaya, “An overview of disaster recovery in virtualization technology,” J. Artificial Intelligence, vol.6, no.1, pp.60–67, 2013.
[16] D. Arsenault, A. Sood, and Y. Huang, “Secure, resilient computing clusters: self-cleansing intrusion tolerance with hardware enforced security (SCIT/HES),” Availability, Reliability and Security, 2007, ARES 2007, The Second International Conference on, pp.343–350, IEEE, 2007.
[17] A.K. Bangalore and A.K. Sood, “Securing web servers using self-cleansing intrusion tolerance (SCIT),” Dependability, 2009, DEPEND’09, Second International Conference on, pp.60–65, IEEE, 2009.
[18] A. Ali-Eldin, J. Tordsson, and E. Elmoorthy, “An adaptive hybrid elasticity controller for cloud infrastructures,” Network Operations and Management Symposium (NOMS), 2012 IEEE, pp.204–212, IEEE, 2012.
[19] J. Lim, Y. Kim, D. Koo, S. Lee, S. Doo, and H. Yoon, “A novel adaptive cluster transformation (ACT)-based intrusion tolerant architecture for hybrid information technology,” The Journal of Supercomputing, vol.66, no.2, pp.918–935, 2013.
[20] P. Sousa, N.F. Neves, and P. Verissimo, “Hidden problems of asynchronous proactive recovery,” Proc. Workshop on Hot Topics in System Dependability, 2007.
[21] P. Sousa, A.N. Bessani, M. Correia, N.F. Neves, and P. Verissimo, “ Highly available intrusion-tolerant services with proactive-reactive recovery,” Parallel and Distributed Systems, IEEE Transactions on,
[22] J. Huang and Q. Ai, “Analysis of a virtualization-based recovery approach for intrusion tolerance systems,” Computer Sciences and Applications (CSA), 2013 International Conference on, pp.41–46, IEEE, 2013.

[23] S. Heo, J. Lim, M. Lee, S. Lee, and H. Yoon, A novel intrusion tolerant system based on adaptive recovery scheme (ARS), vol.215, pp.71–78, Springer, 2013.

[24] S. Malik and F. Huet, “Adaptive fault tolerance in real time cloud computing,” Services (SERVICES), 2011 IEEE World Congress on, pp.280–287, IEEE, 2011.

[25] P. Das and P.M. Khilar, “VFT: A virtualization and fault tolerance approach for cloud computing,” Information & Communication Technologies (ICT), 2013 IEEE Conference on, pp.473–478, IEEE, 2013.

[26] Q. Nguyen and A. Sood, “Quantitative approach to tuning of a time-based intrusion-tolerant system architecture,” Proc. 3rd Workshop Recent Advances on Intrusion-Tolerant Systems, pp.132–139.

[27] Y. Shin, S. Song, Y. Lee, and H. Yoon, “A novel intrusion tolerant system using live migration,” IEEE TRANSACTIONS on Information and Systems, vol.97-D, no.4, pp.984–988, April 2014.

[28] M. Garcia, N. Neves, and A. Bessani, “Diversys: Diverse rejuvenation system,” Proceedings of the INFORUM-Simpósio de Informática, 2012.

[29] M. Jensen, N. Gruschka, and R. Herkenhöner, “A survey of attacks on web services,” Computer Science-Research and Development, vol.24, no.4, pp.185–197, 2009.

[30] C.M. Krishna, “Fault-tolerant scheduling in homogeneous real-time systems,” ACM Computing Surveys (CSUR), vol.46, no.4, pp.1–34, 2014.

[31] C. Wu and R. Buyuya, Cloud data centers and cost modeling, Morgan Kaufmann, 2015.

[32] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, and R. Buyya, “Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” Software: Practice and Experience, vol.41, no.1, pp.23–50, 2011.

[33] M. Alhamad, T. Dillon, C. Wu, and E. Chang, “Response time for cloud computing providers,” Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services, pp.603–606, ACM, 2010.

[34] NSFOCUS Information Technology Co. Ltd., “NSFOCUS mid-year DDoS threat report 2013,” report, 2013.

[35] Lawrence Berkeley National Laboratory, “EPA-HTTP: a day of HTTP logs from the EPA WWW server,” http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html, accessed: June 30, 2015.

Bumsoon Jang is a Ph.D candidate at School of Computing in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. He received the B.S in Computer Science from Korea Military Academy in Seoul, Korea, in 2003 and the M.S. degree in Computer Science from Texas A&M University in 2007. His main researches include the system security, especially the design and analysis of recovery-based intrusion tolerant system using the virtualization technique.

Seokjoo Doo received the B.S degree from the Korea Military Academy, Seoul, Korea, in 1995, the M.S. degree in electronic engineering from Yonsei University, Seoul, Korea, in 1999, and the Ph.D. degree in electrical and computer engineering from The Ohio State University, Columbus, OH, in 2008. In 1999, he joined the Department of Electronic Engineering, Korea Army Academy at Yeong-Cheon, Korea, where he is currently a Professor. His research interests include modeling of nonlinear RF devices as well as network security.

Soojin Lee received the B.S degree in Computer Science from Korea Military Academy in Seoul, Korea, in 1992, the M.S. degree in Computer Science from Yonsei University, Seoul, Korea, in 1996, and the Ph.D. degree in Computer Science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2006. He is currently a Professor of the Department of Computer Science and Engineering, Korea National Defense University (KNDU). His research interests include mobile network security, intrusion detection system, and cyber warfare.

Hyunsoo Yoon received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1979, the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1981, and the Ph.D. degree in computer and information science from The Ohio State University, Columbus, OH, in 1988. He is currently a Professor at School of Computing in KAIST. His research interests include mobile ad hoc networks, wireless networks, and network security.