Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Output difference feedback and system benefit control based dynamic heterogeneous redundancy architecture*

Sisi SHAO[1,7], Zhibo HE[2], Shangdong LIU[3,7,8], Weili ZHANG[4], Fei WU[6,7,8],
Fukang ZENG[3,7], Jun ZUO[3,7], Longfei ZHOU[3,7], Yukun NIU[5], Yimu JI[†‡3,5,7,8]

[1]*School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*School of International Business, Xi'an Jiaotong Liverpool University, Suzhou 215123, China*
[3]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[4]*School of Foreign Languages, Information Engineering University, Zhengzhou 450006, China*
[5]*Purple Mountain Laboratories, Nanjing 211111, China*
[6]*School of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[7]*Institute of High Performance Computing and Big Data, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[8]*Nanjing Center of HPC China, Nanjing 210003, China*
[†]E-mail: jiym@njupt.edu.cn

Received Apr. 2, 2024; Revision accepted Oct. 22, 2024; Crosschecked Aug. 5, 2025

Abstract: Mimic active defense technology effectively disrupts attack routes and reduces the probability of successful attacks by using a dynamic heterogeneous redundancy (DHR) architecture. However, current approaches often overlook the adaptability of the adjudication mechanism in complex and variable network environments, focusing primarily on system security while neglecting performance considerations. To address these limitations, we propose an output difference feedback and system benefit control based DHR architecture. This architecture introduces an adjudication mechanism based on output difference feedback, which enhances adaptability by considering the impact of each executor's output deviation on the global decision. Additionally, the architecture incorporates a scheduling strategy based on system benefit, which models the quality of service and switching overhead as a bi-objective optimization problem, balancing security with reduced computational costs and system overhead. Simulation results demonstrate that our architecture improves adaptability towards different network environments and effectively reduces both the attack success rate and average failure rate.

Key words: Mimic defense; Adjudication mechanism; Scheduling strategy; Executor output difference; System benefit

https://doi.org/10.1631/FITEE.2400251 **CLC number:** TP393

1 Introduction

Cyberspace has long faced the challenge of being vulnerable to attack and difficult to defend, as traditional passive defense technologies (Cho et al., 2020; Jiang et al., 2024) are unable to effectively respond to complex cybersecurity threats (Hu HC et al., 2018; Shao et al., 2023a) due to their reactive and specific nature. To address these issues, active defense, which adopts dynamic, fault-tolerant, and reconfigurable

‡ Corresponding author

* Project supported by the National Key R&D Program of China (Nos. 2023YFB2904004 and 2023YFB2904000), the Jiangsu Key Development Planning Project (No. BE2023004-2), the Natural Science Foundation of Jiangsu Province (Higher Education Institutions) (No. 20KJA520001), and the Postgraduate Research Practice Innovation Program of Jiangsu Province (No. KYCX22_1021)

ORCID: Sisi SHAO, https://orcid.org/0009-0005-9026-0133; Yimu JI, https://orcid.org/0000-0001-7019-3942

techniques to create flexible environments that prevent cyber threats (Wang ZH et al., 2023), has become a hot research topic. As a typical active defense technique, cyber mimic defense (CMD) (Wang YW et al., 2018; Wu JX 2022a, 2022b) introduces dynamic heterogeneous redundancy (DHR) with a negative feedback control mechanism to change the system's execution environment based on adjudication outcomes, improving the detection and attack efforts for attackers (Hu JJ et al., 2024; Rehman et al., 2024). The DHR architecture connects different constructed executors in parallel and adjudicates the independently running executors' outputs to obtain the final result (Ren et al., 2020; Li et al., 2021; Fu et al., 2022), reducing the risk of simultaneous failures in the same functional module. The architecture mainly consists of five modules (Tong and Guo, 2021; Zheng et al., 2022): input/output, processing, adjudication, scheduling, and building, as shown in Fig. 1.

Fig. 1 illustrates that the adjudication and scheduling modules are central to the DHR architecture (Wang ZH et al., 2021; Zhu et al., 2021; Wei et al., 2022), and are crucial for generating external uncertainty. Recent advances in mimic active defense technologies have mainly focused on the adjudication and scheduling modules within the DHR architecture (Lu ZP et al., 2017).

For the adjudication mechanism, Lin SJ et al. (2018) introduced a contest arbitration model based on a multi-mode adjudication algorithm, which considers the efficiency of the system while ensuring the system security. Lu YQ et al. (2021) transformed the adjudication into a fuzzy evaluation process by unifying the analysis of consistency, historical confidence, and heterogeneity to improve the correctness of the adjudication algorithm. For the scheduling strategy, Yu et al. (2022) proposed an executor scheduling algorithm based on runtime, reputation, and switching overhead, which can reduce the average system failure rate and improve the defense capability of network protection devices while ensuring that the system overhead is basically the same. Zhang et al. (2020) quantified executor heterogeneity and proved the reasonableness of the quantization method, enabling better distinction between different executors. Wu T et al. (2021) proposed an improved DHR (IDHR) architecture based on the idea of clustering, which clusters the heterogeneity of executors using the k-medoids algorithm before the scheduling module, and evaluated the security of the proposed architecture in terms of attack success rate and control rate. Lucy (2024) introduced two-phase dissimilarity executor selection algorithms based on recombination techniques to select the executor with the maximum phase dissimilarity quantitative value for the processing module. Liu et al. (2018) proposed an executor scheduling algorithm based on the minimum similarity of random seeds. The algorithm excludes executors whose similarity exceeds a threshold based on randomly selected seed executors to constitute the final scheduling scheme, while considering

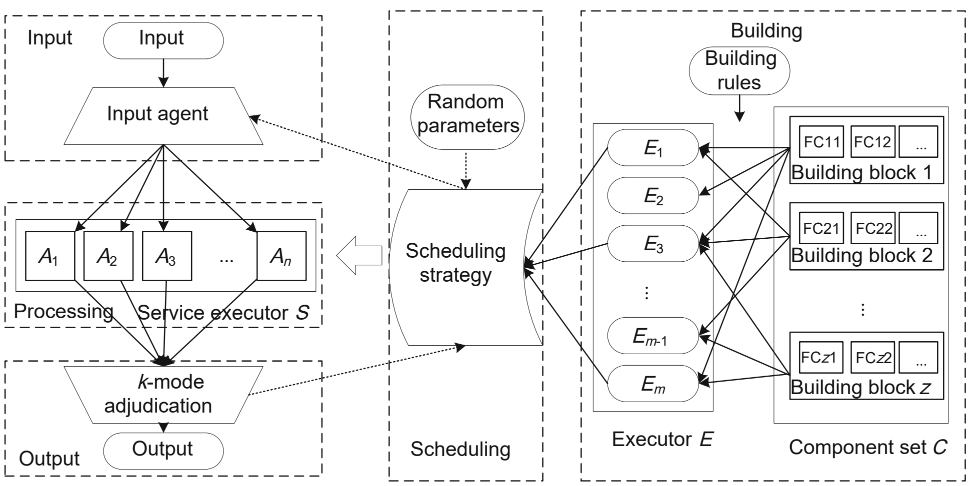

Fig. 1 DHR architecture in mimic defense

the dynamics and heterogeneity of the scheduling algorithm. Compared with other algorithms, it has a longer scheduling period and a lower average failure rate.

Based on the above research, most current adjudication mechanisms focus on improving adjudication accuracy. However, in many complex network environments, computational errors can lead to varying local adjudication results, rendering these mechanisms ineffective. Additionally, current scheduling strategies prioritize system security but often overlook the effects of quality of service and system overhead. To address these issues, we propose a DHR architecture based on output difference feedback and system benefit control. This architecture accounts for output deviations, system overhead, and quality of service while striving to maximize overall system security. The main contributions of this work are as follows:

1. For the adjudication module of the DHR architecture, we propose an adjudication mechanism based on the output difference feedback. This mechanism calculates each executor's output difference by confidence and local adjudication results. It considers the impact of the output deviation of each executor on the global adjudication result, ensuring effective adaptation to network environments with inherent output differences.

2. For the scheduling module of the DHR architecture, we propose a scheduling strategy for executors based on system benefit, which models the quality of service and switching overhead as a bi-objective optimization problem. It is solved via a heuristic algorithm to increase the security gain of the system.

3. Simulations evaluate our architecture across multiple dimensions, comparing system overhead, scheduling period, attack success rate, and failure rate against existing methods.

2 Overview of the architecture

We propose a DHR architecture based on output difference feedback and system benefit control, as shown in Fig. 2. The adjudication module selects the executor with the smallest output difference using a weighted average, considering the impact of each executor's output deviation on the global adjudication result. The negative feedback control module determines the scheduling strategy by calculating the quality of service and switching overhead, considering the benefits and overhead of the system after the introduction of the DHR architecture.

The implementation process of our DHR architecture is as follows:

1. At time t ($t \in T$, T is a period), the input agent receives n copies of the input requests, which are sent to n heterogeneous executors. These executors simultaneously process the requests and send their local adjudication results to the adjudication module.

2. The adjudication module calculates a weighted average $\overline{Y}(t)$ of the local adjudication results for all executors based on the executor confidence and the normalized output difference $e_i(t)$ for each executor i ($i = 1, 2, ..., n$) between its local result and $\overline{Y}(t)$. The result with the smallest $e_i(t)$ is selected as the global adjudication result $Y(t)$, and is sent to the output agent.

3. The negative feedback control module switches the scheduling strategies in two ways: routine and emergency switching. For every time interval Δt, the module calculates the quality of service Q based on the difference $D(t)$, the historical average confidence $H(t)$, and the running overhead C_{run}. It determines the scheduling strategy by combining Q and the switching overhead C_{switch}. If the quality difference between the new and current strategies exceeds the threshold γ and the switching overhead is less than δ, a routine switch occurs. If executor i is under attack or experiences an unknown error, and the output difference $e_i(t)$ exceeds the threshold η, its confidence $h_i(T)$ drops. If $h_i(T)$ falls below the threshold ξ, the negative feedback control module urgently switches to the first scheduling strategy that satisfies the quality of service and switching overhead.

3 Improved adjudication mechanism and scheduling strategy

3.1 Related indicators

3.1.1 Executor confidence

Confidence is an important indicator for assessing the reliability of an executor. In the DHR architecture, higher confidence in the selected executor correlates with greater system reliability. In

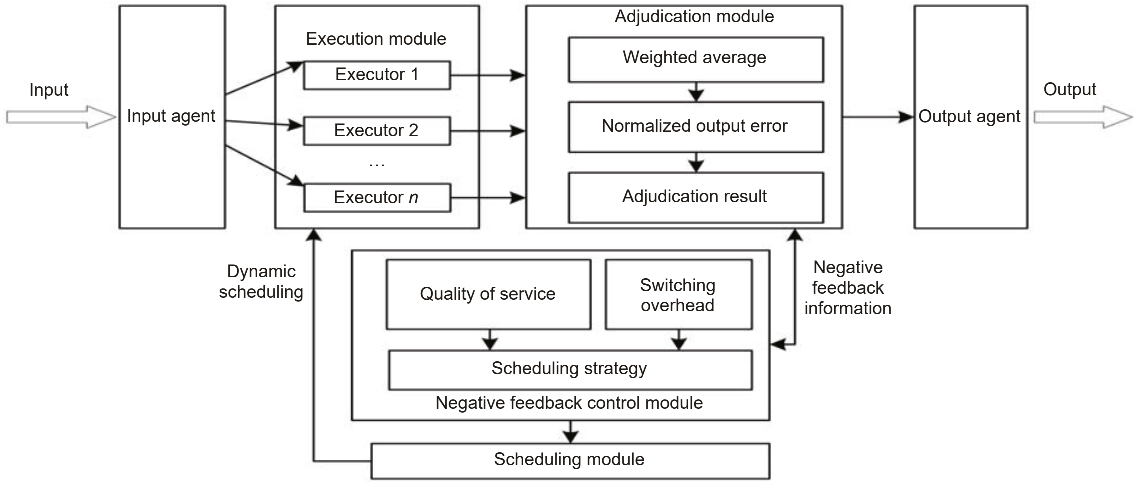

Fig. 2 DHR architecture based on output difference feedback and system benefit control

our architecture, confidence is dynamically updated based on the difference between local and global adjudication results. If the difference is within the threshold η at time t, the executor retains high confidence. If the difference exceeds the threshold η, confidence drops, potentially triggering an emergency switch if $h_i(T)$ falls below the threshold ξ, leading to the removal of low-confidence executors.

Let $y_i(t)$ denote the local adjudication result of executor i at time t, $Y(t)$ denote the global adjudication result provided by the adjudication module at time t, and $\zeta_i(T)$ denote the number of time steps that executor i has been selected by the scheduling strategies during T. So, the historical confidence $h_i(T)$ is calculated by

$$h_i(T) = \frac{\sum\limits_{t\in T} f(|Y(t) - y_i(t)|)}{\zeta_i(T)}, \tag{1}$$

where $f(x) = 1$, if $x \leq \eta$, and $f(x) = 0$, if $x > \eta$.

Let $\boldsymbol{h}(t) = [h_1(t), h_2(t), \ldots, h_n(t)]$ denote the confidence values of all executors (n is the total number of executors) in the execution module at time t. The average confidence $H(t)$ is calculated as follows:

$$H(t) = \frac{1}{n}\sum_{i=1}^{n} h_i(t). \tag{2}$$

At time t, if executor i is not selected by the scheduling strategy, its historical confidence remains unchanged from the previous moment. Conversely, if executor i is selected, its historical confidence is updated dynamically according to the mechanism described as

$$h_i(t) = \begin{cases} \frac{\zeta_i(t-1)h_i(t-1)+f(|Y(t)-y_i(t)|\leq\eta)}{\zeta_i(t-1)+1}, i \in S, \\ h_i\left(t-1\right), i \notin S, \end{cases} \tag{3}$$

where $h_i(0)$ denotes the initial confidence level of executor i and S denotes the set of executors. The executor confidence is updated in real time based on the negative feedback information to reduce the probability of selecting malicious executors during the algorithm's execution and to improve the system's reliability.

3.1.2 Executor heterogeneity

An executor consists of components under different construction rules, and its heterogeneity depends on the differences among these components. Each component contains different blocks with distinct characteristics. Information systems can be divided into different components according to their functions, and we assume the existence of four types of components: code, module, transport, and operation. Each component contains multiple blocks, for example, the code component includes languages such as Java, C++, Python, and C#. The weights of these four component types are denoted by w_1, w_2, w_3, and w_4, with different construction rules assigning varying weights. Therefore, the heterogeneity $d_i(t)$ of executor i can be expressed as

$$d_i(t) = w_1 c_i(t) + w_2 m_i(t) + w_3 t_i(t) + w_4 o_i(t), \tag{4}$$

where $c_i(t)$, $m_i(t)$, $t_i(t)$, and $o_i(t)$ denote the difference of the blocks in the code, module, transport, and operation system components at moment t, respectively. It follows that the average difference $D(t)$ of this scheduling scheme at time t is

$$D(t)=\frac{1}{n}\sum_{i=1}^{n} d_i(t). \tag{5}$$

3.1.3 Running overhead

The system's running overhead mainly consists of four components: C_{proxy}, C_{decision}, C_{schedule}, and C_{executor}. Therefore, the running overhead is obtained by

$$C_{\text{run}}=C_{\text{proxy}}+C_{\text{decision}}+C_{\text{schedule}}+C_{\text{executor}}. \tag{6}$$

We assume that the executors require essentially the same resources to process the same input request, but have different running times; so, the executor processing overhead C_{executor} is expressed as

$$C_{\text{executor}}=\sum_{i=1}^{n} t_i \times C_i, \tag{7}$$

where t_i is the running time of executor i, C_i is the overhead required for executor i to perform computation on the task. Compared to other overheads, the executor processing overhead consumes a significant amount of system resources, which affects the system performance. Therefore, the system's running overhead can be approximated as

$$C_{\text{run}} \approx \sum_{i=1}^{n} t_i \times C_i. \tag{8}$$

3.1.4 Switching overhead

When the negative feedback control module selects a new executor scheduling strategy, the system needs to switch the original scheduling strategy, which incurs a switching cost, denoted as C_{switch}. This cost is mainly divided into the executor's online overhead (the cost required to bring an executor into the execution module) and the executor's offline overhead (the cost required to remove an executor from the execution module). We assume that the overhead for a single executor coming online or going offline is essentially the same, so the switching overhead is calculated as

$$C_{\text{switch}}=\sum_{i=1}^{x} C_{\text{input}}+\sum_{j=1}^{y} C_{\text{output}}, \tag{9}$$

where x is the number of executors being added to the execution module and y is the number of executors being removed. C_{input} is the online overhead of a single executor and C_{output} is the offline overhead of a single executor. To express the objective function of the scheduling algorithm efficiently, the switching overheads for different scheduling strategies need to be normalized.

Define $C_{\text{switch}}^{u,v}$ as the original switching overhead from scheduling strategy u to scheduling strategy v. Then the normalized switching overhead $C_{\text{switch}}^{\prime u,v}$ is obtained by

$$C_{\text{switch}}^{\prime u,v}=\frac{C_{\text{switch}}^{u,v}-\min(C_{\text{switch}}^{u,v})}{\max(C_{\text{switch}}^{u,v})-\min(C_{\text{switch}}^{u,v})}, \tag{10}$$

where the $\max(\cdot)$ and $\min(\cdot)$ represent the maximum and minimum switching overhead, respectively.

3.1.5 Quality of service

The quality of service in this study is determined by $H(t)$, $D(t)$, and C_{run} of the DHR architecture. Higher average confidence and average heterogeneity under scheduling strategy u result in a better quality of service, meaning that Q_u is proportional to $H(t)$ and $D(t)$. Conversely, higher running overhead C_{run} under this scheduling strategy leads to lower system performance and quality of service. Therefore, the quality of service Q_u is calculated as follows:

$$Q_u=H(t)\times D(t)\times\left(\frac{1}{n}\sum_{i=1}^{n} t_i \times c_i\right)^{-1}. \tag{11}$$

3.2 An adjudication mechanism based on output difference feedback

Most current DHR architectures use multi-mode adjudication and consistent adjudication. However, in many network environments, such as state estimation systems in power grids (Jiang et al., 2021), executor outputs may differ, leading to differing local adjudication results. In this case, the aforementioned adjudication mechanism may be inadequate. To address this issue, we propose an adjudication mechanism based on output difference feedback. This mechanism considers the impact of the magnitude of each executor's output deviation on the global adjudication result, making it well-suited for systems with inherent output differences. It enhances the adaptability of adjudication in various complex network application environments.

This adjudication mechanism obtains the weighted average of the output results of n executors, considering the local adjudication results and the confidence levels of the executors, as shown in Eq. (12). The confidence level reflects the reliability of an executor, so we use it as a factor influencing the local adjudication result.

$$\overline{Y}(t)=\frac{\sum\limits_{i=1}^{n} h_i(T)\times y_i(t)}{\sum\limits_{i=1}^{n} h_i(T)}. \tag{12}$$

The normalized output difference between $y_i(t)$ and $\overline{Y}(t)$ is then calculated as follows:

$$e_i(t)=\left|\frac{y_i(t)-\overline{Y}(t)}{\overline{Y}(t)}\right|. \tag{13}$$

Finally, the adjudication mechanism selects the local adjudication result with the smallest output difference as the global adjudication result, which is then sent to the output agent. The adjudication algorithm based on the output difference is detailed in Algorithm 1.

Algorithm 1 Adjudication algorithm for executors based on output difference

Initialize $e_{\min}(t)=1$
for $i=1,2,...,n$
  $\overline{Y}(t)=\frac{\sum_{i=1}^{n} h_i(T)\times y_i(t)}{\sum_{i=1}^{n} h_i(T)}$
  $e_i(t)=\left|\frac{y_i(t)-\overline{Y}(t)}{\overline{Y}(t)}\right|$
  if $e_i(t)<e_{\min}(t)$
    $e_{\min}(t)$=$e_i(t)$
    $\mathrm{ID}_{\min}=i$
  end if
end for
Output the local adjudication result of the executor corresponding to $\mathrm{ID}_{\min}$ as the global adjudication result
for $i=1,2,...,n$
  if $i\in S$
    $h_i(t)=\frac{\zeta_i(t-1)h_i(t-1)+f(|Y(t)-y_i(t)|\leq\eta)}{\zeta_i(t-1)+1}$
  else
    $h_i(t)=h_i(t-1)$
  end if
end for

We theoretically compare the adaptability of the adjudication mechanism proposed in this study with several other adjudication mechanisms across different network environments, as shown in Table 1.

Table 1 Validity of adjudication in different network environments

Method	No difference	Difference
IDHR (Wu T et al., 2021)	✓	–
MD (Lucy, 2024)	✓	–
OMD (Lucy, 2024)	✓	–
RSMS (Liu et al., 2018)	✓	–
Our method	✓	✓

–: the method is not suitable for scenarios with output differences. MD: maximum dissimilarity; OMD: optimal mean dissimilarity; RSMS: random seed & minimum similarity

The results indicate that, unlike our mechanism, the other adjudication mechanisms are not suitable for scenarios where executor outputs contain differences.

3.3 A scheduling strategy based on system benefit

The principle of the DHR architecture is to dynamically add multiple heterogeneous executors to make the system uncertain and secure. Although using multiple heterogeneous executors improves system security, it increases the redundancy, consumes additional system resources, and reduces overall stability (Lin X et al., 2023). To address these problems, we use system benefit to characterize the security gains of the system and propose an executor scheduling strategy based on the benefit. The quality of service and the switching overhead of a mimic active defense system are modeled as a bi-objective optimization problem (Jiang et al., 2023). This optimization problem is solved using a heuristic algorithm to obtain the optimal executor scheduling strategy.

When selecting an executor scheduling strategy, the goal is to maximize system benefit. By definition, system benefit is directly proportional to the quality of service and inversely proportional to the switching overhead. Thus, the objective function after modeling the quality of service and switching overhead is as follows:

$$\begin{aligned}&\text{Max: Benefits}=\lambda_1 Q_u+\lambda_2\frac{1}{C_{\text{switch}}}\\&\text{s.t. }\frac{|Q_u-Q_v|}{|Q_u+Q_v|}\geq\gamma, C_{\text{switch}}^{'u,v}\leq\delta,\end{aligned} \tag{14}$$

where γ and δ are adjusted by historical experience and "Benefits" denotes the system benefit. γ reflects the system's sensitivity, influencing the frequency of scheduling strategy changes, and δ reflects the maximum bumpiness of a single adjustment, with a larger value representing higher system stability. Different network scenarios have varying requirements on the quality of service and system overhead. In scenarios with high quality of service requirements and sufficient system resources, γ and δ can be appropriately increased to ensure system stability. In scenarios with high security requirements, γ and δ can be appropriately reduced to ensure system security.

After obtaining the objective function, the particle swarm optimization (PSO) algorithm is used to efficiently explore the feasible solution space, aiming to quickly find the optimal value of the bi-objective optimization problem. According to the formula of the basic PSO algorithm, the PSO algorithm proposed in our study can be obtained as follows (Shao et al., 2023b; Yadav and Raj, 2024):

$$\begin{aligned}\boldsymbol{V}_i^{k+1} =&\omega k\boldsymbol{V}_i^k + c_1r_1(\boldsymbol{P}_{\text{best},i}^k - \boldsymbol{X}_i^k)\\ &+ c_2r_2(\boldsymbol{P}_{\text{global},i}^k - \boldsymbol{X}_i^k),\end{aligned} \tag{15}$$

$$\boldsymbol{X}_i^{k+1} = \boldsymbol{X}_i^k + \boldsymbol{V}_i^{k+1}, \tag{16}$$

where $\boldsymbol{V}_i$ and $\boldsymbol{X}_i$ are the velocity and position of particle i respectively, c_1 and c_2 are acceleration coefficients typically set to 2, r_1 and r_2 are pseudo-random numbers uniformly distributed in $[0,1]$, $\boldsymbol{P}_{\text{best},i}$ is the best historical position for particle i, $\boldsymbol{P}_{\text{global},i}$ is the best position for all particles, and ω is the inertia weight typically ranging from 0 to 1, balancing the ability of the particle to search globally or locally. Setting a higher ω will promote global search, and a lower ω will promote fast-read local search. In summary, the executor scheduling algorithm based on system benefits is presented in Algorithm 2.

Algorithm 2 Scheduling algorithm for executors based on system benefit

```
if t_now − t_last == Δt
    for v = 1, 2, ..., N /* v is the scheduling scheme and N is the number of possible scheduling strategies in the system */
        H(t) = (1/n) Σ_{i=1}^{n} h_i(t)
        D(t) = (1/n) Σ_{i=1}^{n} d_i(t)
        C_run ≈ Σ_{i=1}^{n} t_i × C_i
        Q_u = H(t) × D(t) × ((1/n) Σ_{i=1}^{n} t_i × c_i)^{−1}
        C_switch = Σ_{i=1}^{x} C_input + Σ_{j=1}^{y} C_output
    end for
    The optimization problem is obtained by modeling the bi-objective optimization problem according to Eq. (14)
    The optimization problem is solved by the PSO algorithm, and the result obtained is used as the switching strategy
end if
if h_i(T) < ξ
    Select the scheduling strategy with the highest system efficiency for switching
end if
```

4 Experiments and analysis

4.1 Experimental settings

We assume that there are $m = 10$ functionally equivalent but structurally heterogeneous executors in the executor pool, and that the executors remain in standby mode, consuming minimal system resources. The system margin n (i.e., the number of executors running simultaneously in the system) is a prerequisite for executing the DHR architecture. Theoretically, a higher system margin increases system security and performance consumption. In practice, a system redundancy of $n = 3$ is typically sufficient to meet the requirements and effectively defend against attackers. To verify system gains under different system margins, we set the execution margin of the DHR system to $n = 3, 4, 5$ with a time interval $\Delta t = 10$ s. The initial confidence level of each executor is set to 0.5, that is, $\boldsymbol{h}(0) = [0.5, 0.5, ..., 0.5]$. The lower confidence limit ξ is set to 0.25, the minimum acceptable quality of service γ is set to 0.3, and the maximum switching overhead of the scheduling strategy δ is set to 0.8. Regarding the variability of executor components, we ignore the differences among the template components and focus solely on the variability in the code, transport, and operation components. The weights w_1, w_3, and w_4 are each set to 1/3. The differences among the component bodies are shown in Table 2.

We first evaluate our architecture in terms of average confidence, average heterogeneity, scheduling randomness, and system benefits. Then we compare our method with the IDHR (Wu T et al., 2021), maximum dissimilarity (MD) (Lucy, 2024), optimal mean dissimilarity (OMD) (Lucy, 2024), and random seed & minimum similarity (RSMS) (Liu et al., 2018) methods, focusing on the average scheduling period, attack success rate, and average failure rate. The IDHR method first divides the executors into

Table 2 Differences between component bodies

Component	Type	Difference
Code	Java	0.20
($w_1 = 1/3$)	C++	0.40
	Python	0.60
	C#	0.80
Transport	TCP/IP	0.20
($w_3 = 1/3$)	IPX/SPX	0.35
	NatBRUI	0.50
	JSON	0.65
	Binary	0.80
Operation	VMware Ubuntu	0.20
($w_4 = 1/3$)	VirtualBox Ubuntu	0.35
	VMware CentOS	0.44
	VirtualBox CentOS	0.56
	VMware Windows	0.68
	VirtualBox Windows	0.80

subclasses based on the heterogeneity degree, and then uses a randomized scheduling algorithm to select the executors. The MD and OMD methods consider the heterogeneity among executors and select executors with the longest difference distance and the best average difference distance, respectively, to form the final scheduling strategy. The RSMS method also considers the difference, excluding those with a difference below a certain threshold from the execution module by random seeding.

4.2 Experimental analysis

In this subsection, we first demonstrate the effectiveness of the proposed method from three aspects: average confidence, scheduling randomness, and system benefits. We then compare our method with other methods in four aspects: average scheduling period, system overhead, attack success rate, and average failure rate.

4.2.1 Verification of validity

1. Average confidence. The executor confidence reflects the system's reliability and is the basis of whether the system can execute properly under occasional or malicious attacks. The average confidence of the DHR system is determined by the executor confidence selected into the execution module, which is calculated in Eq. (2). A higher average confidence indicates greater overall system reliability. The scheduling strategy proposed in this study considers multiple indicators and prioritizes the selection of executors with higher confidence levels for the execution module. If an executor is attacked, resulting in an adjudication error exceeding a predefined threshold, the confidence level of that executor is reduced. An emergency switch is triggered when it falls below the lower limit of the historical confidence threshold ξ. We calculate the change in the DHR system's average confidence under both attack and non-attack conditions for different system margins, as illustrated in Fig. 3.

In Fig. 3a, our method successfully maintains the DHR system's average confidence at a high level while considering other metrics. As the number of scheduling events increases, the average confidence of the DHR system under different margins increases and stabilizes. This is because executors have not been attacked and their confidence levels increase dynamically during the adjudication process. However, as scheduling continues, the confidence levels stabilize according to the dynamic updating mechanism, resulting in an average confidence level of around 0.95.

As shown in Fig. 3b, during the early stage of the experiment, the system is not attacked, and the average confidence steadily increases and eventually stabilizes. However, when the system is attacked, the average confidence of the system decreases significantly according to the dynamic update mechanism for confidence, which reduces the historical confidence of the compromised executor. In response, the negative feedback control module performs executor switching based on the feedback information. The compromised executors are removed, re-entering the pool of heterogeneous executors after being reset. The new scheduling strategy then restores and stabilizes the upward trend in the DHR system's average confidence.

2. Scheduling randomness. Scheduling randomness is measured through the average number of online executors and the average working time for each executor, serving as an important indicator of whether the executors are fully used. Additionally, scheduling randomness reflects the security of the DHR architecture; in other words, the higher the randomness, the greater the security. We conducted 100 simulations at different system margins, calculating the average number of online executors and the average working time for each executor over 100 time instants, as shown in Fig. 4

From Fig. 4, it is evident that similar results are observed across all system margins. The scheduling

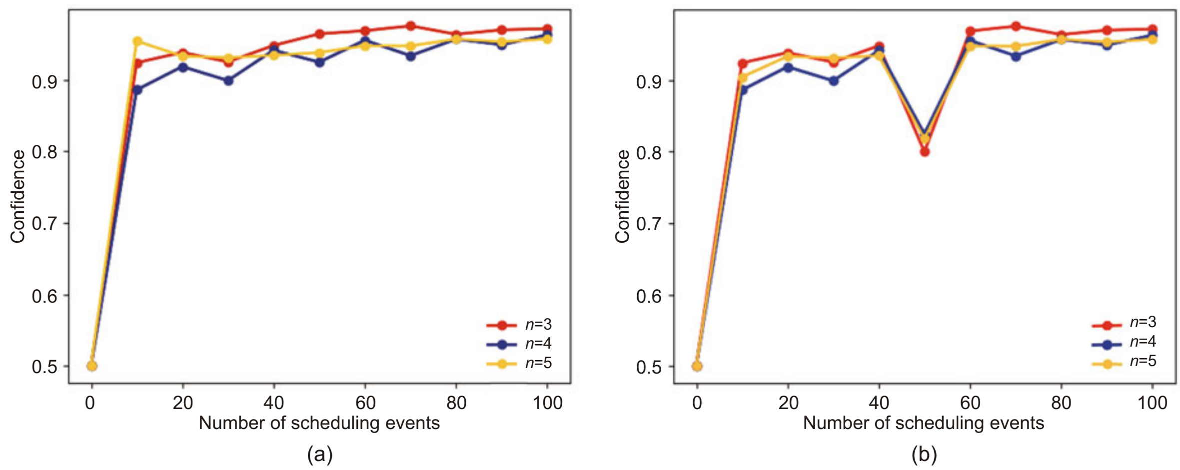

Fig. 3 Average confidence under non-attack (a) and under attack (b)

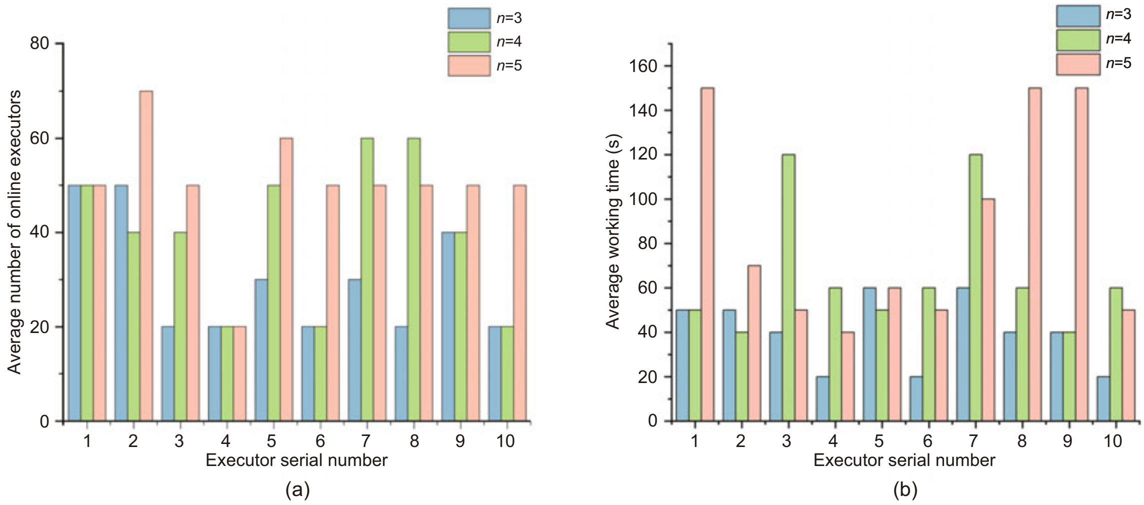

Fig. 4 Scheduling randomness: (a) average number of online executors; (b) average working time of executors

strategy determined by the improved algorithm does not result in only a few executors being repeatedly invoked. Instead, each executor has a chance of being selected by the scheduling strategy. This demonstrates that our method prevents the scheduling strategy from falling into a local optimum and maintains good randomness, ensuring the full utilization of all executors. For an attacker, this randomness makes it difficult to consistently target the same executor over time, thereby reducing the possibility of a successful attack.

3. System benefits. The DHR architecture often sacrifices system performance to improve security. While aiming to improve security, it is crucial to minimize computational costs and system overhead to maximize system benefits. In this study, system benefits are determined by the quality of service and switching overhead, both of which are modeled. The PSO algorithm is used to find the balance between the quality of service and switching overhead, improving the robustness and the security of the system. We analyze the changes in the DHR system benefits over 100 s with different system margins, as shown in Fig. 5.

It can be seen from Fig. 5 that system efficiency gradually increases over time and stabilizes at the fifth scheduling. This shows that the scheduling algorithm proposed in our study effectively balances the quality of service and system overhead, with both high security and robustness. Simulation results show that when $n = 4$, the scheduling strategy determined by our algorithm brings the highest

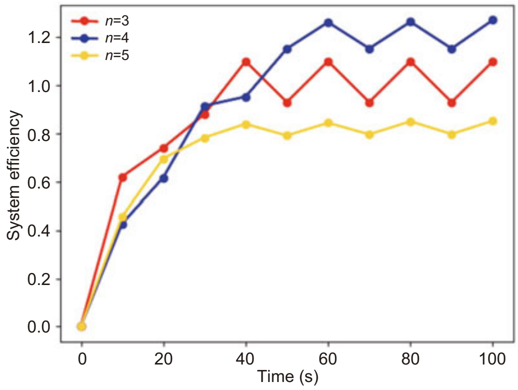

Fig. 5 System benefits with different margins

system benefits. This illustrates that there is no direct correlation between system benefits and the margin. Therefore, in scenarios where system performance is not ideal, choosing a system margin of $n = 3$ is sufficient for the system to effectively defend against attackers while achieving higher system gains.

4.2.2 Validation of comparisons

1. Average scheduling period. The average scheduling period refers to the time interval between scheduling strategies, which is one of the criteria to evaluate the dynamics of the scheduling strategy in the DHR architecture. Ideally, the negative feedback control module can infinitely select different executors for the execution module. However, due to the limited number of executors, the system margin is finite, leading to repeated selections and an established average scheduling period. A longer average scheduling period indicates greater system dynamics. We calculate the average scheduling period of several algorithms with different system margins, as shown in Table 3.

As shown in Table 3, the average scheduling period of our method is longer than that of others. For the IDHR method, all executors are grouped into categories according to the degree of heterogeneity. While this increases the heterogeneity within the processing module, it significantly reduces the number of available scheduling strategies. For the MD and OMD methods, once the scheduling strategy is determined, it remains static, resulting in a scheduling period of 1.00 s for both. The RSMS method considers heterogeneity when determining the scheduling strategy. Although this causes some loss in the system dynamics, it reduces the probability of co-modeling vulnerability and improves the system security. Our method comprehensively considers four indicators, namely confidence, heterogeneity, quality of service, and system overhead, to improve the system's average scheduling cycle and thus leads to greater robustness.

Table 3 Average scheduling period of several algorithms with different system margins

Method	Average scheduling period (s)		
	$n = 3$	$n = 4$	$n = 5$
IDHR	8.40	12.60	23.40
MD	1.00	1.00	1.00
OMD	1.00	1.00	1.00
RSMS	7.31	5.08	3.49
Ours	12.00	33.00	58.20

2. System overhead. As discussed in Section 3.1, the system overhead includes running overhead and switching overhead. To evaluate whether our method effectively reduces the computational cost of the mimic active defense system, we conducted simulations with a system margin of $n = 3$, comparing the system overheads of several methods, as shown in Fig. 6.

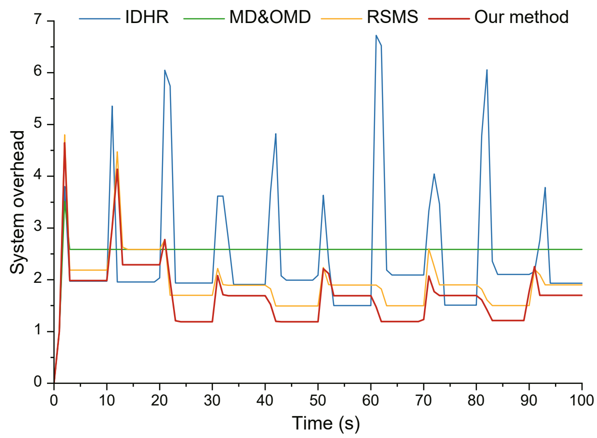

Fig. 6 System overheads under several methods. References to color refer to the online version of this figure

Fig. 6 illustrates that our method consistently maintains lower system overhead compared to other methods, both during the runtime phase and the executor switching phase. The IDHR method uses a random scheduling strategy, leading to a high switching overhead when the executors are switched. In contrast, the MD and OMD methods are static, with no switching overhead once the scheduling strategy is set, resulting in a smooth, straight line representing their system overhead, which is determined solely by running costs. Although the RSMS performs well in terms of system overhead, it is still slightly less efficient than our method. These results suggest that the scheduling strategy proposed in this study effectively balances the various metrics of the mimic active defense system, including quality of service and switching overhead.

3. Attack success rate. The system attack success rate is defined as the probability that an attacker succeeds in launching ε attacks against the system. It is assumed that an attacker relies on a priori information about the system, and an attack cannot succeed without such information. The more a priori information the attacker gathers, the higher the probability of a successful attack. If the attacker consistently receives the same information as the a priori information with each probe, the information accumulates, increasing the attack's chances of success; otherwise, the a priori information fails. Therefore, the attack success rate is related to the number of attacker probes. A lower attack success rate indicates better attack resistance of the system. We simulated the changes in the attack success rate for several methods as the number of probes increases, as shown in Table 4.

Table 4 shows that our method achieves a lower attack success rate compared to IDHR, MD, OMD, and RSMS. This is primarily because the average scheduling period significantly influences the attack success rate. A clear correlation is shown between the system's attack success rate and the average scheduling period. Our method, along with IDHR and RSMS, considers the dynamic and heterogeneous nature of scheduling strategies, resulting in much lower attack success rates than the MD and OMD methods. Our method also considers system benefits, which increase the security gain of the system. In contrast, the MD and OMD methods lack dynamism, allowing the attacker to accumulate experience over time, which increases the probability of successful attacks.

4. Average failure rate. The average system failure rate reflects the probability that the system fails to execute a service correctly under attack, indirectly indicating the system's inability to function properly. We assume that the failure rate of an executor is distributed between 0 and 1 and consider two scenarios of equal ($\alpha_j = 0.1$) and unequal ($\alpha_j \in (0, 0.1)$) failure rates among heterogeneous executors. Keeping other parameters constant, we simulated the average failure rates of several methods across different system margins for both scenarios, as shown in Table 5. To better illustrate the differences in average failure rates among the methods, we analyze our simulation results using the p-value, a statistical measure used to test the significance of a hypothesis, as presented in Fig. 7.

As shown in Table 5, similar results can be observed for both equal and unequal failure rate scenarios. The average failure rate of our method is lower than that of IDHR and RSMS. This is because IDHR and RSMS consider fewer metrics to effectively improve the reliability of the whole system. While the average failure rate of our method is higher than that of MD and OMD in some cases, its attack success rate is significantly lower. Additionally, Fig. 7 shows that the p-value for each method is

Table 4 Attack success rate under several methods

Number of probes	Attack success rate				
	IDHR	MD	OMD	RSMS	Our method
1	0.058 90	0.379 00	0.379 00	0.054 30	0.031 60
2	0.059 30	0.519 00	0.519 00	0.054 30	0.032 50
3	0.059 30	0.570 00	0.570 00	0.054 30	0.032 50
4	0.059 30	0.589 00	0.589 00	0.054 30	0.032 50
5	0.059 30	0.596 00	0.596 00	0.054 30	0.032 50
6	0.059 30	0.598 50	0.598 50	0.054 30	0.032 50
7	0.059 10	0.599 40	0.599 40	0.054 30	0.032 50
8	0.059 10	0.599 80	0.599 80	0.054 30	0.032 50
9	0.059 10	0.599 90	0.599 90	0.054 30	0.032 50
10	0.059 10	0.599 97	0.599 97	0.054 30	0.032 50

Table 5 Average failure rate under several algorithms

Method	Equal failure rate		
	$n=3$	$n=4$	$n=5$
IDHR	2.2000×10^{-3}	8.0227×10^{-5}	1.8134×10^{-4}
MD	3.6751×10^{-4}	1.3350×10^{-5}	6.4437×10^{-5}
OMD	1.3000×10^{-3}	8.4503×10^{-5}	1.6772×10^{-4}
RSMS	7.1909×10^{-4}	2.3656×10^{-5}	9.2103×10^{-5}
Our method	1.6448×10^{-4}	2.0879×10^{-5}	1.4268×10^{-5}
Method	Unequal failure rate		
	$n=3$	$n=4$	$n=5$
IDHR	7.6647×10^{-4}	1.5152×10^{-5}	3.5186×10^{-5}
MD	1.1419×10^{-4}	8.8765×10^{-7}	6.7820×10^{-6}
OMD	3.5989×10^{-4}	1.0583×10^{-5}	1.1182×10^{-5}
RSMS	2.7664×10^{-4}	3.8064×10^{-6}	2.5745×10^{-5}
Our method	1.4400×10^{-5}	1.8190×10^{-6}	1.2340×10^{-5}

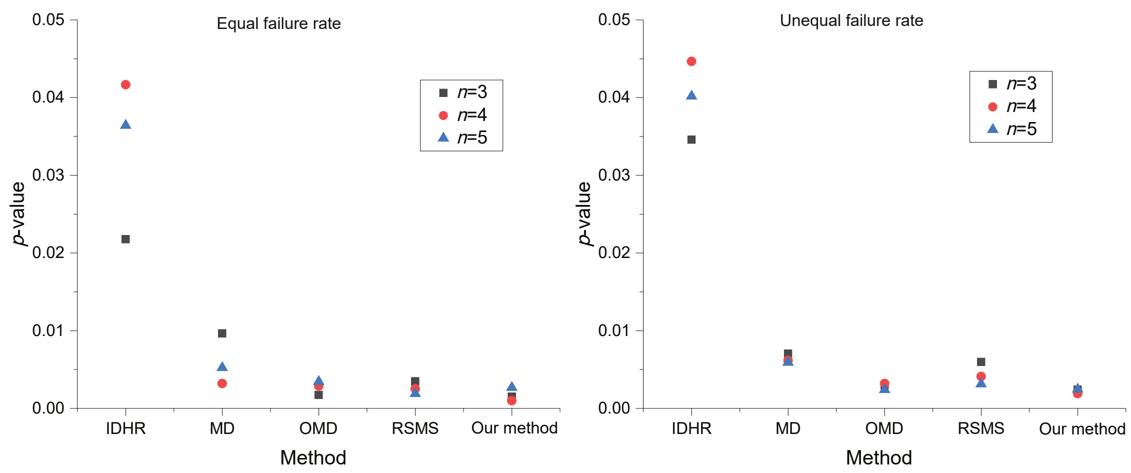

Fig. 7 Significance analysis

lower than 0.05, with our method having the lowest p-value among all. This indicates that our method performs significantly better than the others. These results demonstrate that our method achieves strong performance in terms of average scheduling period, system overhead, attack success rate, and average failure rate.

5 Conclusions

To address the limitations of traditional mimic defense techniques in adapting to different network environments and defending against complex and evolving attacks, we propose an output difference feedback and system benefit control based DHR architecture. This architecture considers the impact of each executor's output differences on the global adjudication result, enabling effective adaptation to varying network environments. Additionally, the architecture introduces metrics such as the heterogeneity, switching overhead, and quality of service to improve the system's defense against attacks. Simulation results demonstrate that the proposed method has high system benefits while maintaining a low attack success rate and average failure rate.

Contributors

Sisi SHAO designed the research. Zhibo HE, Fukang ZENG, Jun ZUO, and Longfei ZHOU processed the data. Sisi SHAO drafted the paper. Weili ZHANG, Fei WU, and Yukun NIU helped organize the paper. Shangdong LIU and Yimu JI revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Cho JH, Sharma DP, Alavizadeh H, et al., 2020. Toward proactive, adaptive defense: a survey on moving target defense. *IEEE Commun Surv Tut*, 22(1):709-745. https://doi.org/10.1109/COMST.2019.2963791

Fu T, Zhen W, Yang F, et al., 2022. Mimic defense equivalent scheduling algorithm based on service quality and credit. IEEE 6th Information Technology and Mechatronics Engineering Conf, p.414-419. https://doi.org/10.1109/ITOEC53115.2022.9734532

Hu HC, Wu JX, Wang ZP, et al., 2018. Mimic defense: a designed-in cybersecurity defense framework. *IET Inform Secur*, 12(3):226-237. https://doi.org/10.1049/iet-ifs.2017.0086

Hu JJ, Li Y, Li ZZ, et al., 2024. Unveiling the strategic defense mechanisms in dynamic heterogeneous redundancy architecture. *IEEE Trans Netw Serv Manag*, 21(4):4912-4926. https://doi.org/10.1109/TNSM.2024.3387725

Jiang DD, Wang ZH, Huo LW, et al., 2021. A performance measurement and analysis method for software-defined networking of IoV. *IEEE Trans Intell Transp Syst*, 22(6):3707-3719. https://doi.org/10.1109/TITS.2020.3029076

Jiang DD, Wang F, Lv ZH, et al., 2023. QoE-aware efficient content distribution scheme for satellite-terrestrial networks. *IEEE Trans Mob Comput*, 22(1):443-458. https://doi.org/10.1109/TMC.2021.3074917

Jiang DD, Wang ZH, Wang Y, et al., 2024. A blockchain-reinforced federated intrusion detection architecture for IIoT. *IEEE Int Things J*, 11(16):26793-26805. https://doi.org/10.1109/JIOT.2024.3406602

Li GS, Wang W, Gai KK, et al., 2021. A framework for mimic defense system in cyberspace. *J Signal Process Syst*, 93(2):169-185. https://doi.org/10.1007/s11265-019-01473-6

Lin SJ, Liu QR, Wang XL, 2018. Competitive arbitration model for mimic defense system. *Comput Eng*, 44(4):193-198 (in Chinese). https://doi.org/10.3969/j.issn.1000-3428.2018.04.031

Lin X, Wu J, Li JH, et al., 2023. Heterogeneous differential-private federated learning: trading privacy for utility truthfully. *IEEE Trans Depend Secur Comput*, 20(6):5113-5129. https://doi.org/10.1109/TDSC.2023.3241057

Liu QR, Lin SJ, Gu ZY, 2018. Heterogeneous redundancies scheduling algorithm for mimic security defense. *J Commun*, 39(7):188-198 (in Chinese). https://doi.org/10.11959/j.issn.1000-436x.2018124

Lu YQ, Huang JX, Cheng Z, et al., 2021. A multi-index mimic voting algorithm based on improved AHP-FCE model. *J Beijing Univ Posts Telecommun*, 44(2):8-13 (in Chinese). https://doi.org/10.13190/j.jbupt.2020-105

Lu ZP, Chen FC, Cheng GZ, et al., 2017. Towards a dynamic controller scheduling-timing problem in software-defined networking. *China Commun*, 14(10):26-38. https://doi.org/10.1109/CC.2017.8107630

Lucy W, 2024. Algorithms and adjudication. *Jurisprudence*, 15(3):251-281. https://doi.org/10.1080/20403313.2023.2243712

Rehman Z, Gondal I, Ge MM, et al., 2024. Proactive defense mechanism: enhancing IoT security through diversity-based moving target defense and cyber deception. *Comput Secur*, 139:103685. https://doi.org/10.1016/j.cose.2023.103685

Ren Q, Wu JX, He L, 2020. Performance modeling based on GSPN for cyberspace mimic DNS. *Chin J Electron*, 29(4):738-749. https://doi.org/10.1049/cje.2020.05.001

Shao SS, Ji YM, Zhang WL, et al., 2023a. A DHR executor selection algorithm based on historical credibility and dissimilarity clustering. *Sci China Inform Sci*, 66(11):212304. https://doi.org/10.1007/s11432-022-3635-2

Shao SS, Liu SD, Li K, et al., 2023b. LBA-EC: load balancing algorithm based on weighted bipartite graph for edge computing. *Chin J Electron*, 32(2):313-324. https://doi.org/10.23919/cje.2021.00.289

Tong Q, Guo YF, 2021. A comprehensive evaluation of diversity systems based on mimic defense. *Sci China Inform Sci*, 64(12):229304. https://doi.org/10.1007/s11432-020-3008-1

Wang YW, Wu JX, Guo YF, et al., 2018. Scientific workflow execution system based on mimic defense in the cloud environment. *Front Inform Technol Electron Eng*, 19(12):1522-1536. https://doi.org/10.1631/FITEE.1800621

Wang ZH, Jiang DD, Wang F, et al., 2021. A polymorphic heterogeneous security architecture for edge-enabled smart grids. *Sustain Cities Soc*, 67:102661. https://doi.org/10.1016/j.scs.2020.102661

Wang ZH, Jiang DD, Lv ZH, 2023. AI-assisted trustworthy architecture for industrial IoT based on dynamic heterogeneous redundancy. *IEEE Trans Ind Inform*, 19(2):2019-2027. https://doi.org/10.1109/TII.2022.3210139

Wei D, Xiao L, Shi L, et al., 2022. Mimic web application security technology based on DHR architecture. Proc Int Conf on Artificial Intelligence and Intelligent Information Processing, p.118-124. https://doi.org/10.1117/12.2660317

Wu JX, 2022a. Cyberspace endogenous safety and security. *Engineering*, 15:179-185. https://doi.org/10.1016/j.eng.2021.05.015

Wu JX, 2022b. Development paradigms of cyberspace endogenous safety and security. *Sci China Inform Sci*, 65(5):156301. https://doi.org/10.1007/s11432-021-3379-2

Wu T, Hu CN, Chen QN, et al., 2021. Defense-enhanced dynamic heterogeneous redundancy architecture based on executor partition. *J Commun*, 42(3):122-134 (in Chinese). https://doi.org/10.11959/j.issn.1000-436x.2021022

Yadav D, Raj BAA, 2024. An efficient swarm intelligence algorithm for multi-objective task scheduling optimization in the context of cloud computing. Int Conf on Automation and Computation, p.148-152. https://doi.org/10.1109/AUTOCOM60220.2024.10486073

Yu F, Liu K, Geng YY, et al., 2022. Multi executor decision algorithm and scheduling algorithm based on differential distance feedback. *Appl Res Comput*, 39(5):1437-1443 (in Chinese). https://doi.org/10.19734/j.issn.1001-3695.2021.10.0462

Zhang JX, Pang JM, Zhang Z, 2020. Quantification method for heterogeneity on web server with mimic construction. *J Softw*, 31(2):564-577 (in Chinese). https://doi.org/10.13328/j.cnki.jos.005615

Zheng Y, Li Z, Xu XL, et al., 2022. Dynamic defenses in cyber security: techniques, methods and challenges. *Digit Commun Netw*, 8(4):422-435. https://doi.org/10.1016/j.dcan.2021.07.006

Zhu ZB, Liu QR, Liu DP, et al., 2021. Research progress of mimic multi-execution scheduling algorithm. *J Commun*, 42(5):179-190 (in Chinese). https://doi.org/10.11959/j.issn.1000-436x.2021072