Journal homepage: https://itce.vn.ua/en

Information Technologies and Computer Engineering **Vol. 23, No. 2. 2026**

Article's History: Received: 12.02.2026 Revised: 27.05.2026
Accepted: 25.06.2026 Published: 03.08.2026

ISSN 1999-9941; e-ISSN 2078-6387
UDC 004.7
DOI: 10.31649/vitce/2.2026.109

Development of an automation algorithm for restoring the operational capability of information and communication networks

Dmytro Vyshnevskyi*
Full-time Listener
Ivan Kozhedub Kharkiv National Air Force University
61023, 77/79 Sumska Str., Kharkiv, Ukraine
https://orcid.org/0009-0007-8666-2951

Vladyslav Baizan
Full-time Listener
Ivan Kozhedub Kharkiv National Air Force University
61023, 77/79 Sumska Str., Kharkiv, Ukraine
https://orcid.org/0009-0008-8334-7961

Dmytro Kalinovskyi
PhD
Ivan Kozhedub Kharkiv National Air Force University
61023, 77/79 Sumska Str., Kharkiv, Ukraine
https://orcid.org/0000-0003-3184-6458

Artem Samokish
PhD
Ivan Kozhedub Kharkiv National Air Force University
61023, 77/79 Sumska Str., Kharkiv, Ukraine
https://orcid.org/0000-0003-1924-9351

Abstract. The aim of this study was to develop and evaluate the effectiveness of an algorithm for the automated recovery of network nodes in information and communication networks using virtualisation and automation tools. An analysis of existing approaches to network equipment redundancy, in particular the Virtual Router Redundancy Protocol and Hot Standby Router Protocol, was conducted, and their key limitations were identified: the inability to restore the router to its full functional state, limited scalability, and reliance on the human factor. It was determined that the average time for manual restoration of network nodes ranges from 10 minutes to 1 hour or more, depending on many factors and conditions. A comparative analysis of modern configuration management tools, including Ansible, Puppet, Chef, and SaltStack, was conducted. The analysis results justified the choice of Ansible, a specialised automation tool, due to its agentless architecture, use of the secure Secure Shell protocol, and "PUSH" data transfer model, which provided instant responses without prior configuration of target nodes. An algorithm for automated deployment of a virtual router analogue in the Proxmox VE hypervisor environment, when the special Zabbix monitoring software detects a physical equipment failure, has been developed. Centralised management of the recovery process has been implemented via the Jenkins automation server, which receives WebHook signals from Zabbix and initiates the execution of Ansible scripts. The algorithm included sequential stages: failure detection, creation of a new virtual machine, application of the current configuration from the backup storage, and verification of the health of the restored node. To quantify effectiveness, an additive time model and a recovery-acceleration intensity coefficient were applied. It was found that the proposed automated algorithm reduces the network recovery time by 8,5 times compared to the manual method from 30-60 minutes to 3-4 minutes. The proposed algorithm was applicable as a supplementary redundancy layer

Suggested Citation:
Vyshnevskyi, D., Baizan, V., Kalinovskyi, D., & Samokish, A. (2026). Development of an automation algorithm for restoring the operational capability of information and communication networks. *Information Technologies and Computer Engineering*, 23(2), 109-120. doi: 10.31649/vitce/2.2026.109.

*Corresponding author (*dimas.sekso@gmail.com*)

within existing information and communication networks for military and civilian purposes, requiring no substantial changes to the current infrastructure

Keywords: network automation; process automation; network infrastructure; virtualisation; network recovery; fault tolerance

Introduction

The development and research of methods for backup and restoration of network control elements capable of ensuring the continuity of information and communication networks is an urgent task. The development and evaluation of approaches for redundancy and restoration of network control elements represent an urgent operational requirement, as the continuous functionality of information and communication systems is of critical importance for modern information and combat operations. Existing protocols, such as Virtual Router Redundancy Protocol (VRRP) (Nadas, 2010) and Hot Standby Router Protocol (HSRP) (Li *et al.*, 1998), provide only logical redundancy. However, these protocols do not fully restore the router's overall functionality or internal logic, thereby significantly limiting their effectiveness in complex, mission-critical network environments. The implementation of more advanced methods to maintain operational readiness and rapidly recover communication infrastructure remains a priority. In particular, Verizon (n.d.) sets strict time limits in its service-level agreements for responding to critical incidents and outages within 15 minutes, aiming to maintain full network availability. This underscores an industry-wide trend toward minimising infrastructure downtime through automated alerts and rapid response. For its part, Otava (n.d.) clearly defines a recovery time objective ranging from 1 to 4 hours, depending on the criticality of the systems. This approach is based on the use of disaster recovery as a service (DRaaS) solutions, where recovery times are directly tied to the provider's financial guarantees. This demonstrates that in modern network environments, restoring functionality within a few hours is a basic standard for system viability.

Thus, existing approaches provide only partial network redundancy and do not solve the problem of fully restoring the functional state of the router, which contradicts the requirements for system fault tolerance and the capabilities of existing solutions. These contradictions are directly manifested at the stage of practical restoration of network performance, where the manual method of replacing equipment is mainly used. The urgency of the problem is confirmed by the D. Donnellan & A. Lawrence (2024), according to which network issues are the leading cause of outages across all IT services, while human error contributes to 66-80% of all downtime incidents. In the absence of clearly defined regulatory recovery time indicators, the actual duration of network downtime directly depends on the human factor and organisational conditions, which is a critical drawback during the performance of combat missions.

In modern research, automation of network infrastructure is considered as a mechanism for increasing the efficiency of administration. In particular, in a study by D.A. Vyshnevskyi & D.O. Kalinovskyi (2025), a method of automated routing recovery on virtualisation using the Proxmox VE platform and the Ansible automation tool was proposed. It was found that the combination of virtualisation and automation enables rapid deployment of a virtual router in the event of a physical equipment failure and implements the principles of Network Function Virtualisation (NFV). The research developed this approach by developing a full-fledged algorithm with a detailed architecture of the recovery system and a quantitative assessment of the efficiency using an additive time model. A.N. Hidayat & W.R.E. Andi (2025) investigated the use of Ansible to automate the deployment of a server environment and found that the use of an agentless architecture reduces operational costs and increases the manageability of virtualised systems. The authors emphasised that the "PUSH" management model enables rapid changes without requiring additional specialised software on target nodes, which is important in closed or restricted network environments. A comparative analysis of configuration management tools was provided in the study by H. Katariya *et al*. (2025), which investigated the capabilities of Ansible, Puppet, SaltStack, and Chef. The results showed that Ansible is more appropriate for environments with a small to medium number of nodes due to its ease of implementation and lack of agents. However, in large distributed infrastructures, performance limitations may arise due to the sequential nature of task execution.

S. Wågbrant & V. Dahlén Radic (2022) conducted an experimental comparison of network device configuration automation using Ansible, Puppet, and SaltStack. The results of the study showed that the agentless approach simplifies the initial deployment of the management system, but requires a clear organisation of scripts to ensure the stability of repeated operations. The authors also noted that automation allows to significantly reduce the time for configuring network parameters compared to the manual method. Therefore, automation will indeed significantly reduce the time for configuring network parameters compared to the manual method. The practical aspects of applying automation in network environments are considered in the study by M.F. Mohd Fuzi *et al*. (2021), which focuses on automating Enhanced Interior Gateway Routing Protocol (EIGRP) network configuration using Ansible. Experimental results showed a significant reduction in configuration time for network devices, from 5,797 seconds with a manual method to 120 seconds with an automated approach. Thus, the obtained data confirm the significant potential of automation tools for increasing the efficiency

of network infrastructure operation, however, the issue of full recovery of a network node in the event of a physical equipment failure was not considered.

The study by O.S. Yeremenko *et al*. (2025), who performed a comparative analysis of the VRRP and GLBP protocols that ensure continuous network operation in the event of a failure of the main gateway, was considered. As a result, it was found that the VRRP protocol is more effective for fast switching between active and backup gateways compared to GLBP. However, this method does not account for the possibility of dynamically deploying virtual router analogues that would fully replace a physical router, subject to additional conditions in the presence of security policies for internal information resources and other components. The study by Ya. Savchenko & S. Levchenko (2026), which investigated the requirements for local area network management software, found that an effective monitoring system should automatically detect failures and anomalies in real time. The authors confirmed that centralised monitoring systems can significantly reduce incident response time and increase the reliability of network infrastructure. Existing works focus on automating configuration or infrastructure deployment, while the issue of fully automated recovery of a network node in the event of physical equipment failure remains under-researched. The aim of the work was to increase the reliability and continuity of the functioning of information systems by developing an algorithm for the backup and restoration of routing elements, which ensures not only the redirection of received traffic, but also the restoration of complete network logic in the event of physical equipment failures.

Materials and Methods

During the work, an experimental research approach was applied to evaluate the efficiency indicators of automated restoration of Information-Centric Networking (ICN) element functionality in the event of a physical failure of network equipment. The chosen research methodology involved comprehensive modelling of an emergency situation by simulating the failure of the main communication node, the role of which was played by the MikroTik hAP AC RB962UiGS-5HacT2HnT hardware router (Riga, Latvia), with subsequent initialisation of the algorithm for automated deployment of a backup virtual router analogue and fixing the time indicators of network recovery. The experimental environment was deployed on a dedicated server node, enabling the virtual router to function within the existing information and communication network. The hardware component was powered by the Proxmox Virtual Environment hypervisor (Vienna, Austria), which acted as a software virtualisation platform.

The implementation of control influences and automated management mechanisms was entrusted to the Jenkins automation server and the Ansible orchestration system, while real-time failure detection was handled by the Zabbix monitoring software. To ensure functional equivalence between the physical and virtual nodes, MikroTik Cloud Hosted Router was chosen, a cloud version of the RouterOS operating system that supports a full set of routing functions in a virtualisation environment and acts as a Virtual Network Function (VNF) running on a standard server platform running the Proxmox VE hypervisor instead of specialised hardware, according to the European Telecommunications Standards Institute standard (ETSI GS NFV 002, 2014). The test network topology included a main router and a client segment to verify service availability. The health status was continuously monitored, with node availability checked via Internet Control Message Protocol (Arkin, 2001) and Simple Network Management Protocol (Case *et al*., 1990) protocols. When the configured triggers of the monitoring system were met, the recovery script was automatically launched via the WebHook mechanism to the Jenkins server.

To quantify the efficiency of the automated recovery process, an additive time model (Hyndman & Athanasopoulos, 2021) was applied, where the total recovery time T is defined as the sum of the sequential, non-overlapping stage durations (1). Additionally, a recovery acceleration intensity coefficient K was introduced, defined as the ratio of manual to automated process execution time according to O.V. Kukhareva & S.A. Kukharev (2009):

$$T = t_{detect} + t_{clone} + t_{conf}, \quad (1)$$

where, t_{detect} – failure detection; t_{clone} – virtual machine cloning; t_{conf} – configuration application.

To quantify the efficiency of the automated recovery process, a recovery acceleration intensity coefficient (K) is introduced, defined as the ratio of the manual process execution time to the automated execution time. The additive time model for calculating the total recovery duration is formalised according to formula 3, representing the sum of sequential stage durations:

$$K = \frac{t_{manual}}{t_{auto}}, \quad (2)$$

where, t_{manual} – time required for manual restoration; t_{auto} – time, using the restoration method.

The recovery algorithm included sequential execution of operations to record the fact of loss of connection, execution of the Ansible system playbook, deployment of a virtual machine from a unified template, launch of an instance of the MikroTik CHR operating system, and installation of the current configuration of the network node with the subsequent control check of the availability of services. Physical failure was simulated by forcibly disconnecting the router's power supply to assess the system's stability under complete node loss. The main indicator of effectiveness was determined by the total recovery time, defined as the interval from the moment of failure detection to the complete restoration of routing. An additive time model was used to formalise the calculations, given the sequential nature of the execution of technological stages that do not overlap in time. In order to ensure the reliability and

repeatability of the results, a series of experiment iterations were conducted, based on the results of which the arithmetic mean value was calculated. The criteria for the success of the measures taken were the complete restoration of connectivity between network segments, the correct formation of routing tables, and ensuring the continuity of network services.

Results and Discussion

Analysis of automation tool selection. Taking into account the requirements for the minimum network recovery time, reducing the impact of the human factor on the quality of recovery, and ensuring information security, there is a need to choose an automation tool that allows you to quickly execute recovery scenarios without prior configuration of target nodes, does not require the installation of additional software, and provides centralised process management. In this case, the use of a tool that ensures the execution of automation algorithms in a limited network environment and a closed network equipment architecture becomes particularly important. Given the above requirements, the Ansible automation platform was chosen to support the research, as its architectural features and principles can meet the requirements without modifying the end nodes. This automation tool is an orchestration and automation platform designed to optimise processes and reduce operational complexity. It is distinguished by its operation execution model, namely, it uses the "PUSH" model. The process of functioning of the "PUSH" model compared to the "PULL" model is depicted in Figure 1.

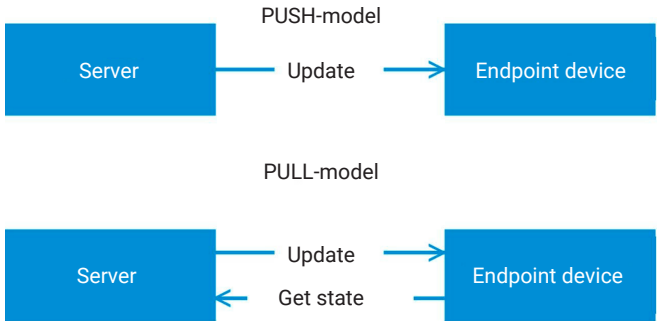

Figure 1. A visual representation of PUSH and PULL models
Source: based on A.N. Hidayat & W.R.E. Andi (2025)

PUSH-model provides instant response efficiency without the need to configure target nodes in advance. The recovery object does not require a return transmission of its state data, since the solution to the problem lies in the clear execution of the recovery algorithm. This approach is fully consistent with the requirements of the Order of the Command of the Signal and Cybersecurity Troops of the Armed Forces of Ukraine No. 369 (2020) and Order of the Command of the Signal and Cybersecurity Troops of the Armed Forces of Ukraine No. 372 (2020), according to which the level of security of information (automated) systems should exclude the possibility of information leakage at all levels of functioning. Minimising two-way data exchange during recovery reduces the number of potential leakage channels, increasing the overall security of the infrastructure.

This automation tool has several advantages that make it the most rational choice for research. The main feature of Ansible is that it does not require the installation of agents on end devices, since it uses a secure SSH connection to operate, to debug, and manage devices, which most often have a closed software architecture. In addition, the use of agent systems used in Puppet, SaltStack and others is impractical due to the inability to install third-party management agents. This automation tool eliminates this limitation by using a cryptographically protected network protocol that provides secure remote management without the need for additional agents. A flexible module system allows you to manage different types of equipment and application programming interfaces (APIs), and support for a large number of ready-made modules speeds up the development and deployment of specific scenarios.

Compared to Chef, Puppet, SaltStack, and other automation tools, Ansible offers several practical advantages, especially for processing small amounts of information and for applications that do not require complex infrastructure. Its scenarios are understandable in YAML, which makes the tool easier to learn when network maintenance personnel change, compared to Chef, which uses Ruby, or Puppet, which uses its own DSL and provides quick orientation in the scenario. Moreover, Ansible uses a PUSH management model, allowing you to immediately apply changes without waiting for agents to run periodically, as with Puppet or Chef. An important difference between Ansible and SaltStack is the features of its architectural management model. Ansible uses an agentless approach to interacting with managed nodes, while SaltStack involves deploying and maintaining agents on each managed node. This approach reduces the number of additional software components in the system and simplifies the process of initial deployment and maintenance of the infrastructure. As a result, using Ansible, it is possible to reduce the requirements for the initial configuration of the environment and reduce the time to implement automation tools, which is a priority within the framework of this study.

The main advantage of Ansible is that it is used not only as a configuration management tool, but also as a full-fledged platform that provides comprehensive process orchestration. Its functionality includes support for a managed, predictable state of system components, guaranteeing the stability and consistency of the infrastructure regardless of its scale. At the same time, it provides a holistic approach to software deployment, allowing you to perform actions in a reproducible, holistic, and controlled manner, which minimises operational risks and reduces downtime. Due to its versatility and wide range of applications, Ansible forms a single, standardised automation system in which you can implement the full life cycle of infrastructure and software changes. This significantly reduces operational complexity, reduces the cost of supporting tools, and unifies approaches to environment management.

Despite the significant advantages, M. Borgenstrand (2018) and H. Katariya *et al*. (2025) noted that one of the shortcomings of the tool is the lack of state preservation between runs. Since Ansible is agentless and usually operates under the PUSH model (Fig. 1), unlike agent systems that store node state and centralised configuration data in a database, it does not store state centrally between runs. This complicates tasks that require constant monitoring of the previous state of the infrastructure without additional workarounds. Despite this drawback, within the framework of the study of network equipment redundancy, this is not critical, since the approach under consideration involves a one-time launch of the recovery scenario, rather than constant management of a large number of nodes in real time. In such cases, constant-state preservation across runs is not required, since task execution does not depend on the history of previous configurations. What is important is the correct execution of the playbook during each run, and this is fully consistent with the agentless nature of Ansible. Thus, the lack of a centralised state preservation mechanism does not affect achieving the set goal. In the sources S. Wågbrant & V. Dahlén Radic (2022) and AWX Project (2024), it was noted that the main problem of this tool is performance in large environments. Analysis of the Ansible architecture, as documented by the tool's developers, shows that its performance depends directly on the number of managed nodes and the parallel execution parameters. At scales of about 300 managed hosts, there is a need to optimise the management infrastructure, in particular by increasing the number of execution nodes or limiting the number of simultaneous tasks. This is explained by the sequential execution of operations and the use of SSH connections, which make Ansible less effective in environments with a large number of nodes than agent-oriented solutions such as SaltStack.

As for the above limitation, it primarily applies to scenarios where the tool is used to manage hundreds or thousands of nodes. In such cases, the sequential task execution model can indeed lead to an increase in the execution time of the scenarios, since not all operations scale effectively or can be optimised for fully parallel processing. At the same time, for the study's implementation, this limitation is not critical, since the network recovery automation system is used only to start a single management node, which does not create a load typical of large distributed environments. Thus, potential performance issues do not affect the solution's effectiveness within the study's framework, as the volume of operations performed and the number of managed nodes remain minimal. Also, there is a problem with the tool's performance in the runtime environment, as it depends on the compatibility of modules with specific platform versions, which sometimes requires updating or refining the scripting algorithm. To justify the choice of Ansible as an automation tool, it was evaluated against five criteria: architecture, work model, flexibility, performance, and reliability. The evaluation results are presented in Table 1.

Table 1. Overall assessment of the Ansible tool

Criterion	Conclusion
Architecture	Agentless, works via SSH or API. Ansible does not create additional points of failure, which is critical for network devices.
Work model	PUSH model. The control node sends the task item execution script to the target devices.
Flexibility	Supports a wide range of modules, you can automate both network devices and various end devices.
Productivity	Optimal for small to medium environments. It may be slower at large scales due to the sequential execution of tasks.
Reliability	High with correct scenario description. Minimum dependencies due to the lack of agents.
Expediency of use	For fault-tolerant networks, it is critical that automation works even when central services are unavailable, so using this tool is quite appropriate and well-suited for rapid, repeatable deployment of virtual network equipment.

Source: based on V. Aditi & A. Kumar (2020), N. Islavath (2020), S. Wågbrant & V. Dahlén Radic (2022), M.D. Elradi (2023), A.N. Hidayat & W.R.E. Andi (2025), B. Meijer *et al*. (2025)

The comparative assessment confirms that Ansible is the most suitable automation tool for the purposes of this study. Its agentless architecture, PUSH execution model, and SSH-based secure communication provide the necessary combination of speed, simplicity, and security for automated network node recovery. Despite performance limitations at large scale, these do not affect the research context, where a single management node is operated.

Research on the efficiency improvement algorithm. Analysis of the study by M.F. Mohd Fuzi *et al*. (2021) shows that, in EIGRP networks, the use of a device configuration

automation tool increased network efficiency. The reduction in configuration and maintenance time, implemented using Ansible, allowed for faster parameter setup and network maintenance. When comparing the time required for manual and automated device configuration, the automated process improved efficiency by reducing the time needed to perform certain network management actions. The results of the study demonstrated that the use of this tool reduced the time required to perform operations from 5,797 seconds under manual configuration to 120 seconds. The indicated results regarding the reduction in configuration time due to the use of Ansible are important to consider during the study. To quantitatively confirm the obtained result, the coefficient of intensity of recovery acceleration can be used, which is defined as the ratio of the process execution time during manual setup to the time of automated execution, and obtain the results according to formula 1 (Kukhareva & Kukharev, 2009):

$$K = \frac{5{,}797}{120} = 48.31,$$

The calculations indicate that the use of the automation tool sped up the configuration and maintenance of network equipment by more than 48 times compared to the manual approach, demonstrating a significant increase in the efficiency of this automation approach. Since the studied recovery algorithm involves performing certain actions, automation of this process allows to minimise time costs and eliminate the probability of errors by service personnel, which most often occur with the manual recovery method. Within the developed environment, automatic collection of network equipment configuration files and an automated router recovery script were implemented using the virtual machine template provided by the Proxmox Virtual Environment (Proxmox VE) virtualisation platform (Lee, 2024), the Ansible automation tool, and the MikroTik Cloud Hosted Router operating system. In this case, the network recovery system should be designed to ensure full autonomy of the recovery process upon detecting a device failure. Moreover, the interaction among the selected system components should be based on the principles of modularity and centralised management, as shown in Figure 2.

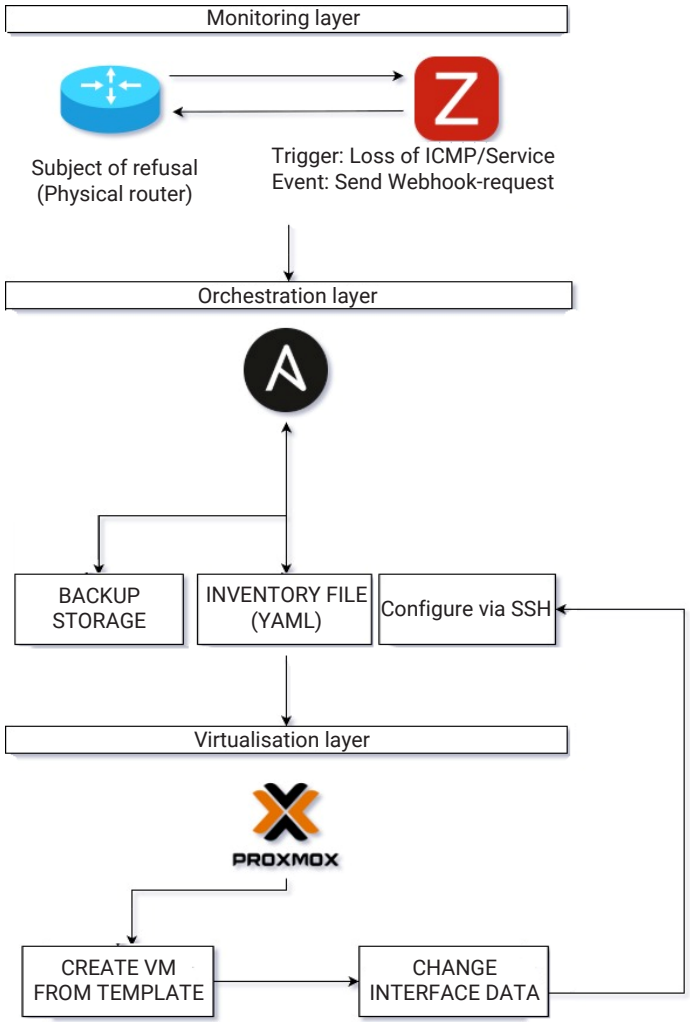

Figure 2. Automated recovery environment architecture

Source: developed by the authors

At the initial stage, the system uses an auxiliary subsystem that automatically records and stores configuration files that record all the basic settings of network equipment, including interface MAC addresses and security keys (if available). This ensures accurate restoration of the router configuration, even when the external network enforces security policies that restrict access to equipment by MAC address. The proposed system meets the requirements for information support of an ACS, which must collect, accumulate, and process data to support the management system. This guarantees the continuity of the information flow, a critical requirement for systems of this class, according to the manual (Ministry of Defence of Ukraine, 2003). During recovery, the collected information is used to correctly assign the original MAC addresses to the newly created virtual machine interfaces, eliminating the possibility of conflicts and connection failures when replacing a physical router with a virtual one. To monitor the status of network equipment, special software Zabbix (Shokhin, 2015) was used, which is one of the most common tools for centralised monitoring of network infrastructure (Gill *et al.*, 2011), and allows not only to obtain information about a specific device but also to add specified conditions in changing indicators that cause the initialisation of a specific event. In the study, the mandatory condition is the loss of communication with specific network equipment, and the event is defined as a call to the recovery scenario for a failed device. Figure 3 demonstrated network topology.

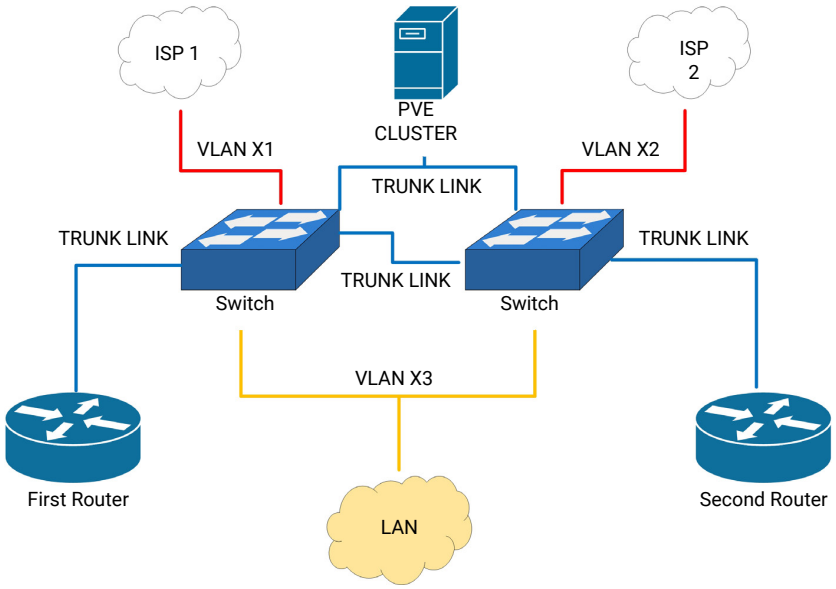

Figure 3. Typical ICM structure

Source: developed by the authors

The given topology reflects the typical structure of an information and communication network using edge routers, a switching layer, and a centralised control node, which allows us to assess the algorithm's performance in combat conditions. To manage the deployment process and ensure continuous integration of network components, the Jenkins automation server (Hyun *et al.*, 2024) was used, which acts as a central link between the monitoring system and orchestration tools. This software receives control signals (WebHook) from Zabbix and initiates the execution of relevant tasks for automated infrastructure formation. The specified interaction of systems implements the principle of event-driven automation (Event-Driven Automation) (Sriramoju, 2023), according to which the detected event is a node failure that automatically initiates the execution of a defined recovery scenario without human intervention. Within the framework of the proposed algorithm, Jenkins is responsible for launching Ansible scripts, thereby eliminating the human factor and ensuring guaranteed repeatability of the network recovery process.

As part of the recovery algorithm, the configuration environment is updated by loading backup configuration files from the backup storage. After performing the specified actions, the system adapts the network interface parameters to the changed operating conditions. As a result of the end of the application of the updated settings, a phased recovery process begins in accordance with the algorithm for checking the compatibility or availability of execution environments, the correspondence of the versions of additional system management tools, and the recovery start algorithm itself. At the final stage, the restored configuration is checked in operating mode, after which the system returns to the normal operating state. In case of errors detected during the control check, the algorithm provides for the initialisation of a procedure to reapply the configuration parameters, ensuring the stability of the recovery process and preventing the system from entering an incorrect

state. On average, the full recovery cycle from the moment the failure conditions are triggered to the restoration of full functionality takes approximately 200-230 seconds, which is significantly less than the time for manual recovery. The general logic of the functioning of the automated network equipment recovery system, which includes the stages of failure detection, virtual analogue deployment, configuration application and health check, is shown in Figure 4.

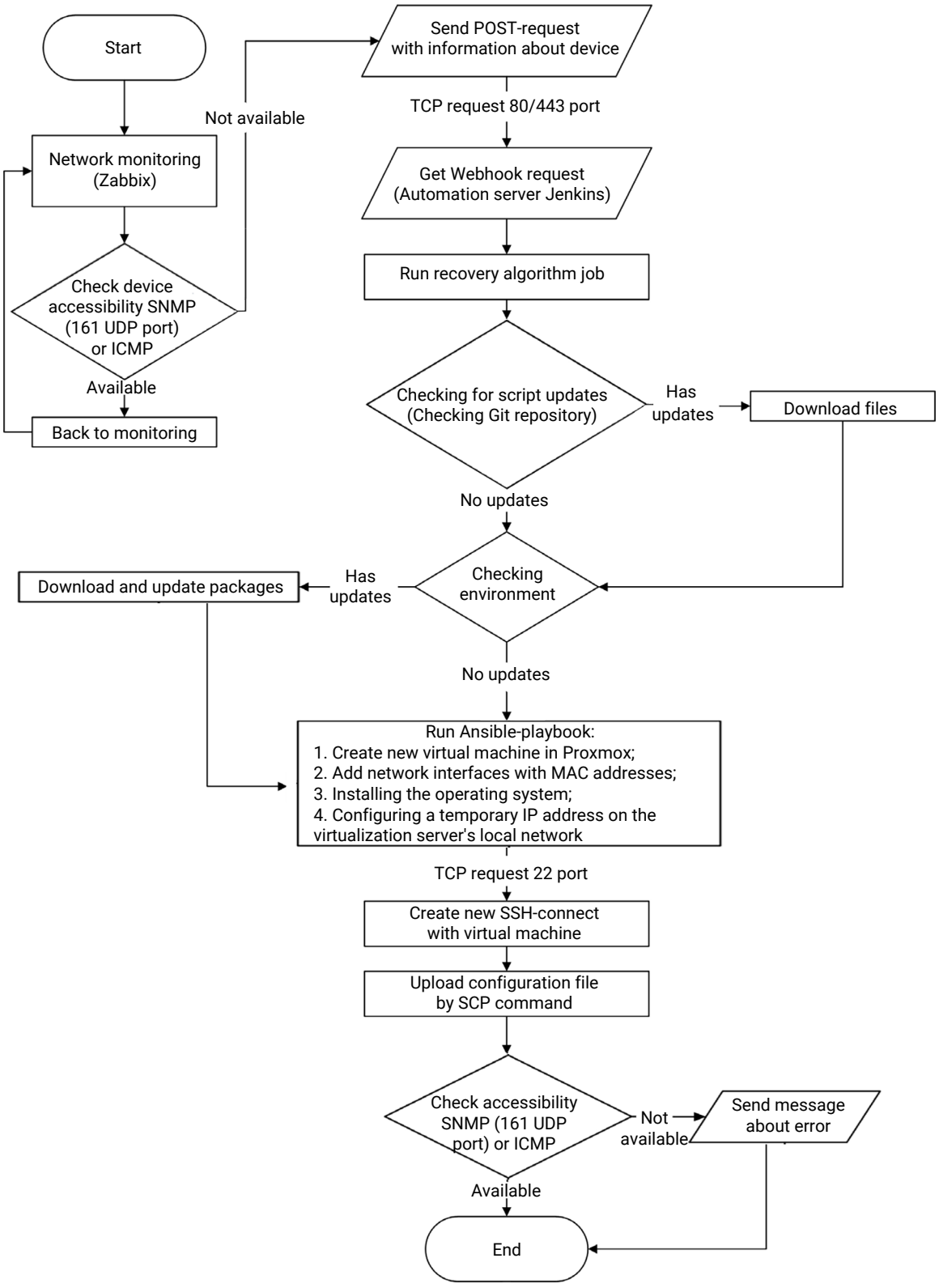

Figure 4. Flowchart of the algorithm for automated recovery of network equipment
Source: developed by the authors

To quantify the effectiveness of the developed automated recovery algorithm, an additive time model is introduced (Hyndman & Athanasopoulos, 2021). The total recovery time is defined as the sum of the durations of the individual stages according to formula 1. Accordingly, the duration of the algorithm execution of the main stages:

$$T = 40 + 155 + 15 = 210 \text{ sec.}$$

Therefore, the full cycle of automated network node recovery takes 210 seconds, or approximately 4 minutes.

According to the result obtained, the acceleration intensity coefficient is calculated using formula 2:

$$K = \frac{1{,}800}{210} = 8.5.$$

Since the acceleration intensity coefficient (K) is 8.5, this demonstrates how much faster the automated process is than the manual one. Given the results, it is possible to conclude that the ICN state recovery time has increased significantly, meeting the standards for assessing the effectiveness of the ACS implementation.

In the first stage of the graphical analysis, the general pattern of the recovery processes was identified, as shown in Figure 5. This diagram illustrates the key stages of a network device's recovery cycle in general terms, showing that the automated method is significantly faster than the manual method at identifying and diagnosing a problem.

To provide a deeper understanding of the mechanism for the automated recovery of network elements, an additional diagram has been created, as shown in Figure 6. This diagram allows for the analysis of the execution of each automated stage, as it breaks down the scenario into specific execution processes, thereby enabling the specific nature of the interaction between stages during critical failures to be taken into account and analysed.

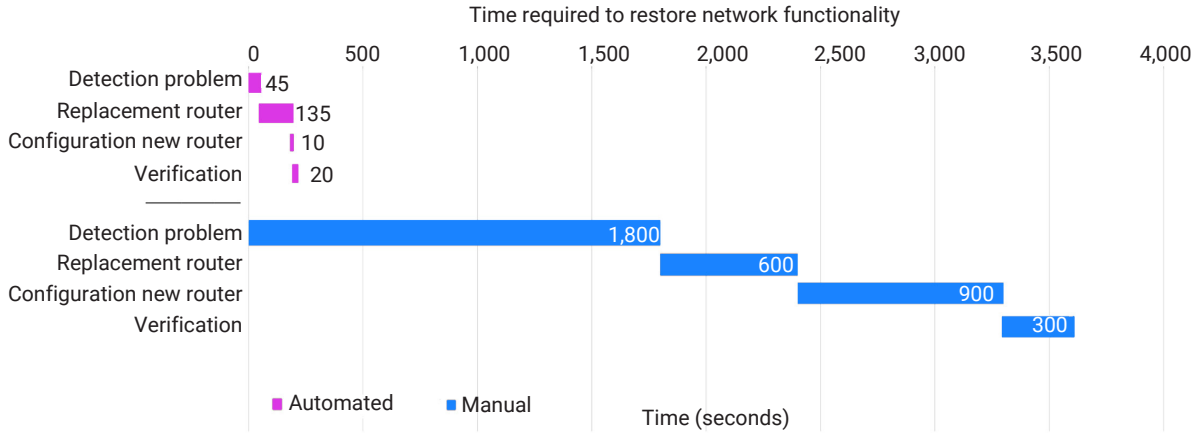

Figure 5. Comparison chart of the time required for manual and automated methods of network equipment recovery
Source: developed by the authors

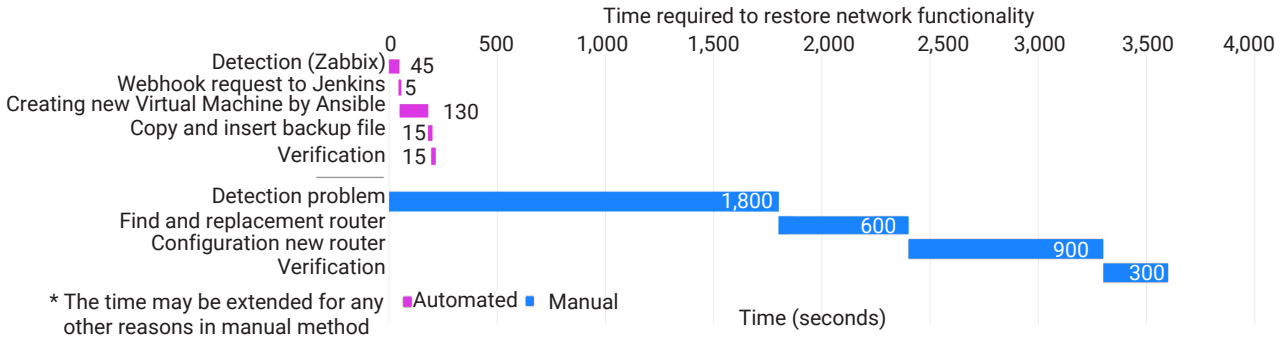

Figure 6. A detailed comparison chart showing the time required for manual and automated recovery of network equipment
Source: developed by the authors

The proposed algorithm therefore represents a practical and measurable improvement over existing manual recovery methods. The integration of Zabbix, Jenkins, Ansible, and Proxmox VE into a unified event-driven recovery pipeline ensures full process autonomy, eliminates human error at critical recovery stages, and provides repeatable and verifiable results across multiple test iterations. The obtained results are consistent with established concepts of network resilience and survivability, where the ability to rapidly restore functionality after failures is considered a critical property of modern communication systems (Sterbenz *et al.*, 2010).

The study confirmed that traditional methods of network equipment redundancy provide only partial system performance and cannot restore the router to its full functional state. This leads to increased network node downtime, reliance on administrator qualifications, and complicates the continuity of operation of information and communication systems, especially in combat conditions, where rapid restoration of communication and access to data is critical. The study showed that the average time for manual restoration of network nodes can range from 10 minutes to an hour, and the duration of coordination and configuration adaptation processes further increases the delay, which does not meet the requirements for the efficiency and reliability of modern ACS. Thus, it can be concluded that this approach does not meet the requirements of timeliness, since the source indicates that the performance of tasks cannot be high-quality without the use of "survivable and protected information (automated) systems".

Accordingly, a study was conducted, during which it was determined that replacing equipment and restoring network performance took approximately 10 minutes to an hour, but these time indicators can increase under the following conditions: lack of an experienced network administrator; up-to-date technical documentation; timely identification of a failed element and other factors. The specified time interval is unsatisfactory for modern ICNs, especially in the conditions of performing combat missions, where the continuity of data exchange, minimal downtime and rapid restoration of full network functionality are of critical importance, which will lead to an increase in decision-making time. Thus, there is a need to implement automated mechanisms for backup and restoration of network control elements that can significantly reduce the recovery time.

Conclusions

Implementing automation and virtualisation tools enables you to create an effective automated recovery system that provides full process autonomy, synchronises critical network equipment parameters, and verifies performance after recovery. Experimental evaluation showed that the automated process reduces recovery time by 8.5 times compared to the manual approach. The developed algorithm reduces the impact of service personnel and increases the network's reliability and resilience to failures. Further improvement of the system, in particular through optimisation of the initial configuration and the implementation of a full-fledged router terminal management mechanism, will allow you to further reduce recovery time and ensure even higher continuity of information and communication system operations.

The key advantage of such an approach the system's ability to provide a solution that allows replacing a physical router with its virtual equivalent, including not only the transfer of network addresses and basic gateway functionality, but also the synchronisation of all critical device parameters. It is assumed that the integration of a centralised management mechanism will allow automating the launch and configuration of a virtual copy of the router, which will reduce network downtime and the impact of maintenance personnel on errors during recovery. Therefore, this approach to deploying a virtual analogue of the router can be considered as an additional layer of redundancy that complements standard protocols.

Prospects for further research include optimisation of the virtual machine deployment stage to reduce total recovery time below 3 minutes, as well as adaptation of the proposed algorithm for network equipment from other manufacturers, including Cisco, Juniper and others, through the development of manufacturer-specific Ansible playbooks. Furthermore, the applicability of the proposed approach to virtualised network environments in data centres, where network functions are implemented as software components within hypervisor infrastructure, represents a promising direction. Additionally, future work may address integration with SDN-based network management frameworks.

Acknowledgements

To the lecturer of the Department of Mathematics and Software of ACS, Faculty of Automated Control Systems and Ground Support of Aviation Flights, Kharkiv National Air Force University, Pavlo Malko, for assistance in providing hardware equipment for conducting the research.

Funding

None.

Conflict of Interest

None.

References

[1] Aditi, V., & Kumar, A. (2025). Optimizing infrastructure management using ansible. *International Journal of Computer Science Trends and Technology (IJCST)*, 13(5), 15-20.

[2] Arkin, O. (2001). *ICMP usage in scanning. The complete know-how*. Retrieved from https://www.cs.dartmouth.edu.

[3] AWX Project. (2024). *Improving AWX performance*. Retrieved from https://docs.ansible.com.

[4] Borgenstrand, M. (2018). *Network automation – the power of Ansible*. (Student's thesis, Mid Sweden University, Sundsvall, Sweden).

[5] Case, J.D., Fedor, M., Schoffstall, M.L., & Davin, J. (1990). *RFC1157: Simple network management protocol (SNMP)*. Retrieved from https://datatracker.ietf.org/doc/html/rfc1157.

[6] Donnellan, D., & Lawrence, A. (2024). *Annual outage analysis 2024: The causes and impacts of IT and data center outages*. Retrieved from https://datacenter.uptimeinstitute.com.

[7] Elradi, M.D. (2023). Ansible: A reliable tool for automation. *Electrical and Computer Engineering Studies,* 2(1), article number 4. doi: 10.58396/eces020104.
[8] ETSI GS NFV 002. (2014). *Network Functions Virtualisation (NFV); architectural framework*. Retrieved from https://www.etsi.org.
[9] Gill, P., Jain, N., & Nagappan, N. (2011). Understanding network failures in data centers: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 conference* (pp. 350-361). New York: Association for Computing Machinery. doi: 10.1145/2018436.2018477.
[10] Hidayat, A.N., & Andi, W.R.E. (2025). Automating server deployment with ansible to improve performance, virtualization, and network security. *Eduvest-Journal of Universal Studies*, 5(8), 10618-10626. doi: 10.59188/eduvest.v5i8.52084.
[11] Hyndman, R.J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). Melbourne: OTexts.
[12] Hyun, G., Oak, J., Kim, D., & Kim, K. (2024). The impact of an automation system built with Jenkins on the efficiency of container-based system deployment. *Sensors*, 24(18), article number 6002. doi: 10.3390/s24186002.
[13] Islavath, N. (2020). Automating infrastructure management: The power of Ansible for DevOps efficiency. *International Journal of Computer Techniques*, 7(2), 1-10.
[14] Katariya, H., Gedam, M., Lolage, R., Patil, S., & Shrivastav, U. (2025). Comparative analysis of configuration management tools: Chef vs. Ansible, SaltStack, and Puppet. *EasyChair Preprint*, 15755.
[15] Kukhareva, O.V., & Kukhariev, S.A. (2009). Research on network element load when varying network parameters with MPLS technology. *System Research and Information Technologies*, 4, 86-91.
[16] Lee, B. (2024). *Proxmox up and running*. Virtualization Howto.
[17] Li, T., Cole, B., Morton, P., & Li, D. (1998). *Cisco Hot Standby Router Protocol (HSRP)*. Retrieved from https://datatracker.ietf.org/doc/html/rfc2281.
[18] Meijer, B., Hochstein, L., & Moser, R. (2025). *Ansible: Up and running*. Sebastopol: O'Reilly Media Inc.
[19] Ministry of Defence of Ukraine. (2003). *Handbook on air defence*. Kyiv: Ministry of Defence.
[20] Mohd Fuzi, M.F., Abdullah, K., Abd Halim, I.H., & Ruslan, R. (2021). Network automation using ansible for EIGRP network. *Journal of Computing Research and Innovation*, 6(4), 61-72. doi: 10.24191/jcrinn.v6i4.237.
[21] Nadas, S. (2010). *Virtual router redundancy protocol (VRRP) version 3 for IPv4 and IPv6 (No. RFC5798)*. Retrieved from https://datatracker.ietf.org/doc/rfc5798.
[22] Order of the Command of the Signal and Cybersecurity Troops of the Armed Forces of Ukraine No. 369 "Instruction 'Information and Automated Management Systems'". (2020, December). Retrieved from https://sprotyvg7.com.ua.
[23] Order of the Command of the Signal and Cybersecurity Troops of the Armed Forces of Ukraine No. 372 "Instruction 'Tactical communication'". (2020, December). Retrieved from https://sprotyvg7.com.ua.
[24] Otava. (n.d.). *Service level agreement*. Retrieved from https://www.otava.com.
[25] Savchenko, Ya., & Levchenko, S. (2026). Analysis of requirements for software for managing local computer networks. *Measuring and Computing Devices in Technological Processes*, 1, 404-411. doi: 10.31891/2219-9365-2026-85-49.
[26] Shokhin, A. (2015). *Network monitoring with Zabbix*. (Bachelor's thesis, Mikkeli University of Applied Sciences, Mikkeli, Finland).
[27] Sriramoju, S. (2023). Optimizing customer and order automation in enterprise systems using event-driven design. *International Journal of Research Publications in Engineering, Technology and Management*, 6(4), 9006-9016. doi: 10.15662/IJRPETM.2023.0604006.
[28] Sterbenz, J.P.G., Hutchison, D., Çetinkaya, E. K., Jabbar, A., Rohrer, J. P., Schöller, M., & Smith, P. (2010). Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8), 1245-1265. doi: 10.1016/j.comnet.2010.03.005.
[29] Verizon. (n.d.). *Service level agreement*. Retrieved from https://enterprise.verizon.com/terms/us/products/internet/sla.
[30] Vyshnevskyi, D.A., & Kalinovskyi, D.O. (2025). Method of automated restoration of routing in information and communication networks based on virtualization. In *Eighth international scientific and technical conference "Computer and information systems and technologies"* (pp. 92-93). Kharkiv: National University of Radio Electronics.
[31] Wågbrant, S., & Dahlén Radic, V. (2022). *Automated network configuration: A comparison between ansible, puppet, and saltstack for network configuration*. (Bachelor's thesis, Mälardalen University, Västerås, Sweden).
[32] Yeremenko, O.S., Savchenko, R.O., Yakovenko, K.O., & Shestopalov, S.O. (2025). Research on reliability and fault tolerance management in infocommunication networks: Modelling and testing of default gateway redundancy protocols. *Information and Communication Technologies*, 5(1), 15-33. doi: 10.23939/ictee2025.01.015.

Розробка алгоритму автоматизованого відновлення працездатності мережевих вузлів в інформаційно-комунікаційних мережах

Дмитро Вишневський

Слухач
Харківський національний університет Повітряних Сил імені Івана Кожедуба
61023, вул. Сумська, 77/79, м. Харків, Україна
https://orcid.org/0009-0007-8666-2951

Владислав Байзан

Слухач
Харківський національний університет Повітряних Сил імені Івана Кожедуба
61023, вул. Сумська, 77/79, м. Харків, Україна
https://orcid.org/0009-0008-8334-7961

Дмитро Каліновський

Доктор філософії
Харківський національний університет Повітряних Сил імені Івана Кожедуба
61023, вул. Сумська, 77/79, м. Харків, Україна
https://orcid.org/0000-0003-3184-6458

Артем Самокіш

Доктор філософії
Харківський національний університет Повітряних Сил імені Івана Кожедуба
61023, вул. Сумська, 77/79, м. Харків, Україна
https://orcid.org/0000-0003-1924-9351

Анотація. Метою дослідження були розробка та оцінка ефективності алгоритму автоматизованого відновлення мережевих вузлів в інформаційно комунікаційних мережах засобами віртуалізації та інструментами автоматизації. Було проведено аналіз наявних підходів до резервування мережевого обладнання, зокрема протоколів Virtual Router Redundancy Protocol та Hot Standby Router Protocol, та встановлено їх ключові обмеження: неможливість відновлення повного функціонального стану маршрутизатора, відсутність масштабованості та залежність від людського фактору. Визначено, що середній час ручного відновлення мережевих вузлів становить від 10 хвилин до однієї години та більше, в залежності від багатьох факторів та умов. Проведено порівняльний аналіз сучасних інструментів конфігураційного управління таких як: Ansible, Puppet, Chef та SaltStack. За результатами аналізу було обґрунтовано вибір спеціального програмного забезпечення автоматизації Ansible завдяки її безагентній архітектурі, використанню захищеного Secure Shell-протоколу та моделі передачі даних “PUSH”, що забезпечує миттєве реагування без попереднього налаштування цільових вузлів. Розроблено алгоритм автоматизованого розгортання віртуального аналога маршрутизатора в середовищі гіпервізора Proxmox VE при виявленні відмови фізичного обладнання спеціальним програмним забезпеченням моніторингу Zabbix. Реалізовано централізоване управління процесом відновлення через сервер автоматизації Jenkins, який отримує WebHook-сигнали від Zabbix та ініціює виконання Ansible-сценаріїв. Алгоритм включає послідовні етапи: виявлення відмови, створення нової віртуальної машини, застосування актуальної конфігурації з резервного сховища та перевірку працездатності відновленого вузла. Для кількісної оцінки ефективності застосовано адитивну часову модель та коефіцієнт інтенсивності прискорення відновлення. Встановлено, що запропонований автоматизований алгоритм дозволяє скоротити час відновлення мережі у 9 разів порівняно з ручним методом з 30-60 хвилин до 3-4 хвилин. Досліджуваний алгоритм може бути використаний як додатковий рівень резервування в існуючих ІКМ військового та цивільного призначення без суттєвої модифікації наявної інфраструктури

Ключові слова: автоматизація мережі; автоматизація процесів; мережева інфраструктура; віртуалізація; відновлення мережі; відмовостійкість