УДК 004.72
DOI https://doi.org/10.32689/maup.it.2024.4.11

Олексій КЛИМЕНКО
аспіранті кафедри комп'ютерних систем, мереж та кібербезпеки, факультету інформаційних технологій, Національний університет біоресурсів та природокористування України, o.klymenko@nubip.edu.ua
***ORCID:** 0009-0005-2590-1803*

СТВОРЕННЯ SELF-HEALING МЕРЕЖІ

Анотація. *У статті розглянуто створення системи самовідновлення працездатності мережі під час збоїв, описано логічну та фізичну схему комунікацій для створення Self-Healing мережі, алгоритм роботи та функціональну схему такої мережі, частково автоматизовано процес комунікації з Інтернет-провайдером, виконано розрахунки з метою визначення, на який час дана система дає змогу пришвидшити процес вирішення аварій.*

Метою роботи *є обмін напрацюваннями щодо автоматизації відновлення мереж після аварій, створення самовідновлювальної мережі на певних ділянках та висвітлення принципів реалізації Self-Healing.*

Методологія дослідження. *За допомогою методів вимірювання та порівняння деякого набору даних перевіряється спроможність даної автоматичної системи моніторингу бути ефективною. Вимірювання – це метод дослідження, за допомогою якого визначається числове значення деякої величини з використанням одиниці вимірювання об'єкта. Порівняння – один із найбільш поширених методів пізнання, який дозволяє встановити подібність та розбіжність об'єктів [1]. Основну роль в такій системі має програма, написана на мові Python, система моніторингу Zabbix, яка контролює стан Інтернет-каналів та запускає програму у випадку аварії, а також камера ESP32CAM, яка значно спрощує процес отримання стану індикації мережевого обладнання постачальника послуг на точках включення замовника. Для надійного збереження та використання секретів використовується SOPS та Age.*

Наукова новизна. *У статті представлено нову систему самовідновлення працездатності мережі у разі проблем на стороні постачальника послуг, розглянуто приклад автоматизованої взаємодії з провайдером для вирішення типових проблем.*

Висновки. *Розглянута схема комп'ютерної мережі з автоматичною системою моніторингу реалізує відмовостійку самовідновлювальну мережу та практично не залежить від обладнання замовника, здатна закривати типові інциденти з мінімальною участю інженера. Згідно розрахункам, час тривалості аварій на об'єктах можна знизити на третину.*

Ключові слова: *self-healing, моніторинг, автоматизація, комп'ютерна мережа, програмування, скрипт, відмовостійкість.*

Oleksii KLYMENKO. CREATING A SELF-HEALING NETWORK

Abstract. *The article considers the creation of a system for Self-Healing of network performance during failures, describes the logical and physical communication scheme for creating a Self-Healing network, the algorithm of operation and the functional diagram of such a network, partially automates the process of communication with the Internet provider, and performs calculations to determine how long this system can speed up the process of resolving failures.*

The purpose of the work *is to share best practices in automating network recovery after disasters, creating a self-healing network in certain areas, and highlighting the principles of Self-Healing.*

Research methodology. *The ability of a given automatic monitoring system to be effective is tested by measuring and comparing a certain set of data. Measurement is a research method that determines the numerical value of a certain value using a unit of measurement of an object. Comparison is one of the most common methods of cognition, which allows to establish similarities and differences between objects [1]. The main role in such a system is played by a program written in Python, the Zabbix monitoring system, which monitors the status of Internet channels and launches the program in the event of an accident, as well as the ESP32CAM camera, which greatly simplifies the process of obtaining the status of the indication of the service provider's network equipment at the customer's switch-on points. SOPS and Age are used to securely store and use secrets.*

Scientific novelty. *The article presents a new system for self-healing of the network in case of problems on the side of the service provider, and an example of automated interaction with the provider to solve typical problems is considered.*

Conclusions. *The considered scheme of a computer network with an automatic monitoring system implements a fault-tolerant self-healing network and is practically independent of the customer's equipment, capable of closing typical incidents with minimal involvement of an engineer. According to calculations, the duration of accidents at facilities can be reduced by a third.*

Key words: *self-healing, monitoring, automation, computer network, programming, script, redundancy.*

Вступ. Вирішення інцидентів, пов'язаних з Інтернет-каналом, займає значну частину часу мережевих інженерів. Автоматизація процесу вирішення таких проблем, особливо за умови, коли філіал адмініструється віддалено та не має ІТ-спеціалістів по місцю, може зменшити загальну тривалість аварій на об'єктах та вивільнити інженерів під інші задачі, що в свою чергу дозволить більш оптимально розподіляти людські ресурси. В даній статті розглянуто автоматизацію мережі в частині реактивного моніторингу.

Постановка проблеми. Велика розподілена комп'ютерна мережа потребує детального налаштування систем моніторингу для відслідковування інцидентів на різних рівнях. Збільшення кількості

логів та попереджень про можливі чи вже такі, що трапились, аварій, розсіює увагу інженерів. Чим більше подій може бути опрацьовано автоматично, тим більш відмовостійкою та надійною стає мережа, інженери вивільняються від рутинних задач.

Self-Healing (самовідновлювальна) мережа необхідна для мінімізації людських зусиль і витрат, пов'язаних із визначенням причин збою в складних системах.

Аналіз досліджень і публікацій. У досліджені [7] чітко визначено значення терміну Self-Healing. Це – властивість, яка дозволяє системі зрозуміти, що вона працює неправильно, і без (або з) втручання людини вносити необхідні корективи, щоб відновити нормальну роботу. Кожна система з властивостями самовідновлення має здатність виявляти, діагностувати та реагувати на збої.

У попередній статті [2] було розкрито декілька прикладів Self-Healing мереж. Проте, варто зазначити, що досліджень в цьому напрямку небагато, хоча вони тривають. Як правило, в таких дослідженнях розглянуто конкретну ділянку мережі, якій за допомогою програмованої складової надано властивість самовідновлювання [5, 7, 10]. Також зазначається, що основною метою інтеграції функцій самовідновлення в будь-яку мережу є підвищення її надійності та зручності обслуговування. Ці атрибути якості традиційно підвищуються в системах самовідновлення.

Виклад основного матеріалу. В даній статті основна увага приділена деяким прикладам автоматизації реактивного моніторингу.

Реактивний моніторинг – спостереження за IT-інфраструктурою та іншими сервісами в режимі реального часу, можливість визначення невідповідності параметрів та вузьких місць роботи кожного компоненту відносно поточних показників [3].

Фізичне підключення комунікацій. Побудова відмовостійкої мережі з точки зору uplinks (висхідні лінії) часто складається з двох маршрутизаторів, які мають окреме підключення до ISP (Internet Service Provider) [8]. Додатково треба зарезервувати комунікації на рівні розподілення. Наприклад, кореневі комутатори зібрані в Stack, від кожного комутатора є підключення до кожного маршрутизатора. Ці декілька підключень між комутатором та маршрутизаторам мають працювати в режимі LAG (Link Aggregation Group), тобто фізичні кабелі об'єднані в одне логічне з'єднання. У разі неполадок з одним з кабелів мережа продовжить працювати через інші. Критично важливо мати резервне живлення, щонайменше ДБЖ (джерело безперебійного живлення), аби обладнання не було чутливим до проблем в електромережі та не перезавантажувалось. Якщо обладнання має декілька блоків живлення, то їх треба підключати до різних ДБЖ (рис.1).

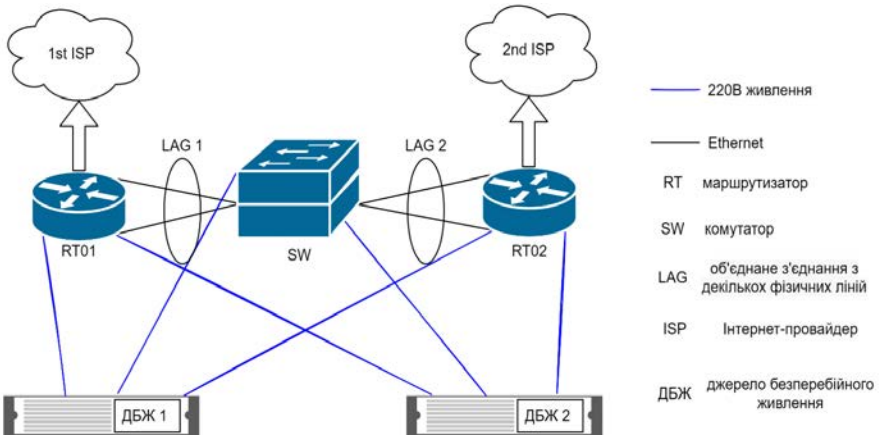

Рис. 1. Базова схема побудови відмовостійкого ядра мережі

Логічна схема комунікацій. Базова логічна конфігурація складається зі стеку технологій IP SLA+TRACK, HSRP (Hot Standby Router Protocol) та EEM (Embedded Event Manager). Перша пара інструментів реалізує перевірку доступності певних ресурсів в мережі Інтернет через підключений Інтернет-канал. Якщо за визначений час на запит не прийшла відповідь, тест IP SLA вважається не пройденим. Запускається лічильник TRACK і, якщо за цей час повторні IP SLA не повернуть позитивний результат, у дію запускається протокол HSRP. На інтерфейсі змінюється пріоритет, і роль головного маршрутизатора перехоплює резервний маршрутизатор. EEM в свою чергу потрібен для очищення NAT-трансляцій аби запобігти переповненню пам'яті, оскільки при перенаправлені трафіку через інший маршрутизатор дані записи втрачають свою актуальність [2].

Моніторинг комунікацій. В цілях моніторингу можна використовувати різні протоколи та системи моніторингу. В даному прикладі розглядається безкоштовна багатофункціональна система моніторингу Zabbix та протокол SNMP, по якому можуть працювати в тому числі пристрої, що були виготовлені раніше. На сервері Zabbix треба додати маршрутизатори. За допомогою шаблона, наприклад, Template SNMP Cisco IP SLA, сервер моніторингу може отримувати дані з маршрутизатора про доступність Інтернет-каналів. На Zabbix треба налаштувати Trigger actions, який буде запускати команду, якщо Інтернет-канал стане недоступним. Ця команда в свою чергу запускає на виконання програму, написану на мові Python, або локально на Zabbix-сервері, або на віддаленому сервері за допомогою Zabbix-agent.

Програмне забезпечення. Програма підключається до потрібного маршрутизатора по SSH (Secure Shell) та виконує команди для діагностики. Якщо з отриманих даних зрозуміло, що проблема на стороні провайдера, програма самостійно підставляє потрібні дані з БД (база даних), такі як ідентифікатор каналу, адреса об'єкту, фото індикації обладнання провайдера, контактні дані адміністратора по місцю тощо та формує лист-заявку в технічну підтримку провайдера. В іншому випадку лист з результатами діагностики надсилається інженерам компанії для подальшого розгляду проблеми. На рисунку 2 зображено частину коду даної програми.

```
59  def get_isp_crd(isp_id, isp_id_range, branch_ip):
60      keys = ["name", "id", "ip", "ip_gw", "location"]
61      params = ["name",
62                f"id_{isp_id_range[isp_id]}",
63                f"ip_{isp_id_range[isp_id]}",
64                f"ip_{isp_id_range[isp_id]}_gw",
65                "location"]
66      values = []
67      for param in params:
68          sops_command = ["sops-v3.8.1.exe", "-d", "-extract",
69                          f"['{branch_ip}']['{param}']",
70                          "c:\\git\\pyneng\\pyneng\\exercises\\isp.yaml"]
71          value = sops(sops_command, branch_ip)
72          if value == "null":
73              err_generate(f"Parameter '{param}' got a value 'null'", branch_ip)
74          else:
75              values.append(value)
76      return dict(zip(keys, values))
77
78
79  def sops(sops_command, branch_ip):
80      value = subprocess.run(sops_command, stdout=subprocess.PIPE,
81                             stderr=subprocess.PIPE, encoding='utf-8')
82      if value.returncode == 0:
83          return value.stdout.strip()
84      else:
85          err_generate(value.stderr.strip(), branch_ip)
```

Рис. 2. Частина коду програми

Захист чутливої інформації. Для підключення до обладнання програмне забезпечення використовує логін та пароль. Крім того, програма взаємодіє з чутливою інформацією про ідентифікатори послуг провайдера, точною адресою точок включення послуг, персональними даними контактів по місцю тощо. Тому вкрай важливо зберігати чутливі дані у зашифрованому вигляді.

З цією метою у даній системі задіяно SOPS+Age. Це система, яка часто використовується у проектах на Git – розподіленої системи керування версіями файлів та спільної роботи. Оскільки файли git проектів розташовуються на публічному або приватному сервері виникає потреба шифрування секретів. Навіть у випадку використання приватного git сервера доступ до нього можуть мати інженери з різними правами [6]. Проте, як в рамках одного файлу надати доступ одному інженеру до однієї ділянки коду, а іншому інженеру – для іншої? Програма Age, використовуючи сучасні алгоритми шифрування (AES 256 GCM), створює криптостійку пару ключів – приватний та публічний. Програма SOPS за допомогою публічного ключа може зашифрувати цілий файл або окремі поля, наприклад, yaml файлу. Публічний ключ зберігається на системі, де буде виконуватись дешифрування, а права доступу до нього надаються тільки відповідному користувачу [9].

Аналогічним чином чутлива інформація для програмного забезпечення може зберігатися на сервері, до якого мають доступ різні адміністратори, у вигляді yaml файлу. Цей файл (або окремі його поля) знаходиться у зашифрованому стані. Право доступу до приватного ключа має лише той адміністратор, який має право запускати на виконання дану програму. Програмне забезпечення в ході своєї роботи запускає програму SOPS для дешифрування чутливої інформації, наприклад, логіну та паролю мережевого обладнання. Далі за допомогою захищеного протоколу SSH відбувається підключення до

мережевого обладнання та взаємодія з ним. Таким чином чутлива інформація не зберігається та не використовується у відкритому вигляді ані в межах системи, ані поза її межами.

Система перевірки індикації обладнання. Досить часто Інтернет-провайдер просить додати фото індикації МК (медіаконвертер). Цей процес також можна автоматизувати. На створення цього рішення надихнув проект «AI-on-the-edge-device» ресурсу GitHub, у якому автор за допомогою камери в автоматичному режимі збирає дані з лічильників води [4]. Для реалізації цього етапу можна використовувати камеру ESP32CAM (рис.3). Для стабільного створення якісних знімків камеру та МК можна з'єднати за допомогою корпусу, розробленого на 3D-принтері. Щоб отримати знімок програма звертається за посиланням до камери. Далі програма аналізує знімок на предмет наявності потрібної індикації на МК та додає цю інформацію до заявки в технічну підтримку провайдера.

Рис. 3. Камера ESP32CAM

Побудова комунікації з Інтернет-провайдером. Оскільки вирішення заявок мають однакові етапи, цей процес також можна автоматизувати. Для цього потрібно домовитись про коди повідомлень з провайдером послуг (таблиця 1). Програма додає потрібний код у лист з міткою, наприклад, «Code:», а відповіді парсить у пошуку аналогічного поля. Коли код-відповідь ідентифіковано, програма на основі цієї інформації приймає рішення про подальші дії. У випадку, якщо надати відповідь без участі людини неможливо, програма переводить запит на інженера. В іншому випадку програма автоматично збирає потрібну інформацію та надсилає відповідь (наприклад, повторна перевірка працездатності Інтернет-каналу).

Приклад послідовності вирішення типової заявки, коли проблема на стороні постачальника послуг:

1. Замовник послуги створює заявку з результатами діагностики.
2. Постачальник послуги підтверджує, що заявку прийнято, та перевіряє інформацію.
3. Постачальник послуги просить повторно перевірити працездатність каналу.
4. Замовник перевіряє роботу Інтернет-каналу та підтверджує, що проблема вирішена.
5. Постачальник закриває заявку.

Таблиця 1

Таблиця кодів

Код	Ініціатор повідомлення	Значення
100	Замовник	Первинний запит (створення заявки)
200	Постачальник	Заявка прийнята
210	Постачальник	Надайте додаткову інформацію
211		Надайте повторно інформацію про індикацію МК
212		Перезавантажте МК
213		Перезавантажте маршрутизатор
214		Уточніть робочі години об'єкту з метою доступу
110	Замовник	Передача додаткової інформації
220	Постачальник	Проблему вирішено. Перевірте працездатність Інтернет-каналу
120	Замовник	Відмова, проблему не вирішено
130	Замовник	Підтвердження, що проблема відсутня. Запит на закриття заявки
230	Постачальник	Заявку закрито

Таким чином більшість заявок може бути вирішено взагалі без участі людини зі сторони замовника, оскільки вони пов'язані з вирішенням типових проблем на стороні провайдера, наприклад, обрив оптичної лінії, відмова кінцевого чи транзитного вузлу тощо. Така автоматизація процесу вигідна в тому числі й постачальнику послуг, тому що дозволяє опрацьовувати заявки швидше та вивільняє додатковий час для їх інженерів.

Алгоритм роботи автоматичної системи моніторингу. На рисунку 4 зображено алгоритм роботи даної системи. Окремо виділено контури IP SLA та Zabbix, які працюють постійно та збирають інформацію про стан мережі. Програма в свою чергу виконується лише за певних підстав.

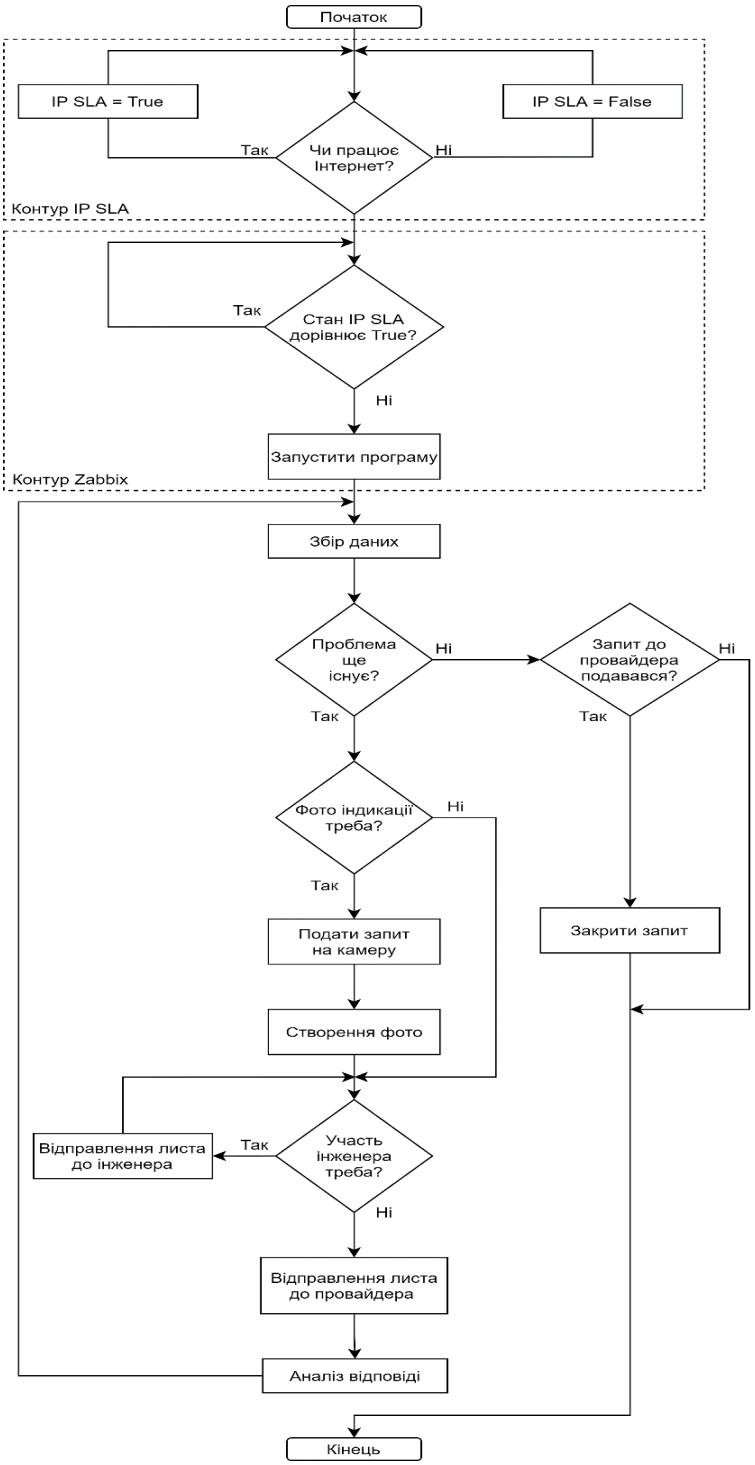

Рис. 4. Алгоритм роботи автоматичної системи моніторингу

Функціональна схема автоматичної системи моніторингу. На рисунку 5 зображено функціональну схему даної системи. Основну роль в ній відіграє програма, написана на мові Python.

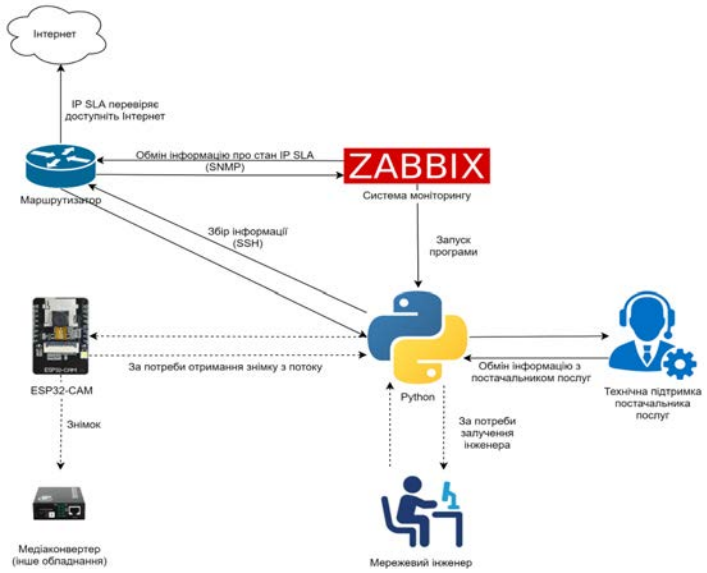

Рис. 5. Функціональна схема автоматичної системи моніторингу

Розрахунки. З метою аналізу, як зазначені вище програмні та апаратні засоби і система взаємодії з Інтернет-провайдером вплинули на час вирішення проблеми, було зібрано ряд даних за останні 6 місяців. В таблиці 2 вказано скільки часу зайняло вирішення проблеми від її початку. У вибірку потрапили лише ті аварії, вирішення яких можна покращити за рахунок перерахованих засобів та систем. Як правило, це обрив ВОЛЗ (волоконно-оптична лінія зв'язку), зависання чи вихід з ладу МК або антени, проблеми на стороні провайдера, що не потребують додаткових дій зі сторони замовника послуг.

Варто зазначити, що на вирішення проблеми мають істотний вплив час реакції на аварію [2], час на уточнення індикації (та\або іншої інформації) та час витрачений на повторну перевірку працездатності сервісу, якщо в результаті виявилось, що сервіс досі не працює.

Таблиця 2

Тривалість аварії

Номер об'єкту	Тривалість аварій на об'єктах за останні 6 місяців						Середній час вирішення заявки (год)	Середній час вирішення заявки (хв)
1	46 год 10 хв	84 год 33 хв	7 год 3 хв				45 год 55 хв	2755
2	24 год 59 хв	3 год 2 хв					14 год 1 хв	841
3	6 год 20 хв	8 год 47 хв					7 год 34 хв	454
4	129 год 6хв	21 год 20 хв	179 год 20 хв				109 год 55 хв	6595
5	239 год 5 хв	19 год 34 хв	21 год 44 хв	6 год 11 хв	191 год 5 хв	165 год 20 хв	107 год 9 хв	6429
6	15 год 31 хв	18 год 51 хв	23 год 51 хв	4 год 40 хв	31 год 20 хв		18 год 51 хв	1131
7	176 год 15 хв	7 год 9 хв	29 год 19 хв	8 год 16 хв	25 год 38 хв		49 год 19 хв	2959
8	24 год 8 хв	4 год 39 хв	36 год	15 год 17 хв			20 год 1 хв	1201
9	19 год 22 хв	28 год 1 хв					23 год 42 хв	1422
10	25 год 32 хв	43 хв	9 год 46 хв	28 год 55 хв			16 год 14 хв	974
11	25 год 6 хв	49 год 48 хв					37 год 27 хв	2247
12	45 год 52 хв	33 хв					23 год 13 хв	1393
13	67 год 15 хв	25 год 17 хв					46 год 16 хв	2776
14	35 хв	6 год 54 хв	3 год 33 хв	13 год 52 хв	3 год		5 год 35 хв	335
15	1 год 29 хв	1 год 17 хв	2 год 1 хв				1 год 36 хв	96
16	81 год 34 хв	15 год 11 хв	20 год 26 хв				39 год 4 хв	2344
17	2 год 10 хв	4 год 43 хв	25 год 24 хв				10 год 46 хв	646
18	2 год 10 хв	1 год 9 хв	8 год 33 хв				3 год 57 хв	237

Таблиця 3

Час реакції

Номер об'єкту	Час реакції на аварії на об'єктах за останні 6 місяців						Загальний час реакції (хв)	Середній час реакції (хв)
1	1 год 44 хв	3 год 21 хв	17 хв				322	107
2	58 хв	17 хв					75	36
3	22 хв	26 хв					48	24
4	8 год 32 хв	24 хв	12 год 37 хв				1293	431
5	10 год 15 хв	19 хв	20 хв	22 хв	9 год 44 хв	2 год 39 хв	1419	237
6	46 хв	25 хв	37 хв	21 хв	52 хв		184	37
7	15 хв	27 хв	2 год 33 хв	44 хв	3 год 13 хв		419	84
8	4 год 9 хв	17 хв	3 год 5 хв	41 хв			492	123
9	57 хв	1 год 14 хв					131	66
10	1 год 17 хв	10 хв	28 хв	53 хв			168	42
11	46 хв	7 год 3хв					469	235
12	2 год 4 хв	15 хв					139	70
13	8 год 3 хв	47 хв					530	265
14	15 хв	24 хв	17 хв	59 хв	35 хв		150	30
15	19 хв	20 хв	18 хв				57	19
16	15 год 10 хв	3 год 17 хв	2 год 13 хв				1240	413
17	25 хв	18 хв	1 год 28 хв				131	44
18	23 хв	16 хв	47 хв				86	29

Таблиця 4

Час на уточнення індикації

Номер об'єкту	Час на уточнення індикації під час аварій на об'єктах за останні 6 місяців						Загальний час уточнення (хв)	Середній час уточнення (хв)
1	1 год 19 хв	2 год 41 хв	0 хв				240	80
2	33 хв	0 хв					33	17
3	0 хв	41 хв					41	21
4	0 хв	1 год 5 хв	2 год 59 хв				244	81
5	5 год 43 хв	0 хв	33 хв	0 хв	4 год 1 хв	6год 14 хв	991	165
6	0 хв	14 хв	29 хв	0 хв	1 год 15 хв		118	24
7	41 хв	0 хв	49 хв	37 хв	0 хв		127	25
8	54 хв	0 хв	1 год 21 хв	0 хв			135	34
9	0 хв	1 год 35 хв					95	48
10	27 хв	0 хв	13 хв	51 хв			91	23
11	44 хв	1 год 33 хв					137	69
12	1 год 19 хв	0 хв					69	35
13	3 год 27 хв	1 год 14 хв					281	141
14	0 хв	31 хв	0 хв	26 хв	0 год		57	11
15	14 хв	10 хв	0 хв				24	12
16	5 год 3 хв	1 год 3 хв	2 год				486	162
17	0 хв	0 хв	49 хв				49	16
18	0 хв	0 хв	34 хв				34	11

У даному випадку час реакції на аварію залежить не тільки від уважності інженера, але й від того, в який час трапилася аварія. Наприклад, якщо це неробочі або вихідні години. Як правило, Інтернет-провайдери мають чергових інженерів NOC (Network Operations Center, центр керування мережею), які працюють цілодобово. Проте, більшість підприємств наймають мережевих інженерів на денну ставку. Крім того, згадані у вибірці об'єкти мають один-два резервних канали, що лишає статусу критичності проблеми з одним з постачальників послуг.

Таблиця 3 показує час реакції на проблему – час між тим, як трапилася аварія на мережі, та тим, коли було направлено листа постачальнику послуг.

В таблиці 4 відображено час, витрачений на уточнення індикації обладнання (наприклад, МК або антени). Тобто ті випадки, коли провайдер запросив інформацію про поточний стан обладнання. Окремо варто зазначити, що дві третини об'єктів підприємства, дані якого взято для вибірки, не мають ІТ-спеціаліста на місці. Це означає, що процес уточнення індикації обладнання часом може займати декілька годин.

Крім того, у семи випадках було витрачено час на повторну перевірку працездатності сервісу, в результаті якої виявилось, що сервіс досі не працює. Сумарно цей час дорівнює 2 години та 7 хвилин.

Керівництвом департаменту ІТ було вирішено, що заявка в Інтернет-провайдер має бути надіслана не раніше 15 хвилин з моменту аварії. У випадку реалізації системи автоматичного моніторингу час реакції на аварію буде фактично сталим. Для цього в Zabbix потрібно вказати необхідний час для виконання команди, яка в свою чергу запускає програму.

У таблиці 3 загальний час реакції для 59 інцидентів дорівнює 7353 хвилин. Якщо б ці інциденти опрацьовувались автоматично, то це би зайняло:

$$T_{автоматичної\ реакції} = 59 * 15 = 885\ (хвилин)$$

У таблиці 4 загальний час на уточнення індикації дорівнює 3252 хвилин. Якби індикація надсилалась у листі-запиті автоматично, то це би не зайняло додаткових хвилин.

Таким чином автоматизована система моніторингу дозволила би заощадити наступну кількість хвилин:

$$T_{заощаджено} = T_{звичайної\ реакції} - T_{автоматичної\ реакції} + T_{уточнення} + T_{повторної\ перевірки} =$$
$$= 7353 - 885 + 3252 + 127 = 9847\ (хвилин)$$

Що складає майже третину часу від сумарної тривалості усіх аварій:

$$9847 / 34835 = X / 100\%$$
$$X = (9847 / 34835) * 100 = 28.27\%$$

Крім того, швидкість вирішення інцидентів має підвищити зменшення часу очікування на відповідь постачальником послуг зі сторони замовника, оскільки такі відповіді частково можуть бути автоматизовані, про що зазначено у пункті «Побудова комунікації з Інтернет провайдером».

Висновки. Розглянута схема комп'ютерної мережі з автоматичною системою моніторингу реалізує відмовостійку самовідновлювальну мережу.

Об'єкт з такою мережею продовжить працювати, якщо:

1. Наявні проблеми з електроживленням.
2. Обривається один з кабелів комунікацій.
3. Виникає аварія з роботою Інтернет-каналу.
4. Виходить з ладу один маршрутизатор або комутатор рівня ядра чи розподілення мережі.

Автоматизований реактивний моніторинг здатен без участі інженера закривати типові інциденти.

Згідно розрахункам, час тривалості аварій на об'єктах можна знизити на третину.

Ключову роль у такій системі має програмне забезпечення, що виконує збір даних, опрацьовує їх та приймає необхідні рішення. Така програма може бути написана на Python або інших мовах програмування, що додає гнучкості системі – інженер може виконувати автоматизацію мережі за аналогічною схемою в тому числі, коли він володіє іншою мовою програмування. Крім того, сервер моніторингу має вміти виконувати або запускати віддалено програми, а мережеве обладнання – відслідковувати стан Інтернет-каналів, що реалізовано у більшості розробників. Таким чином, розглянута автоматична система є масштабованою, гнучкою та фактично незалежною від розробка мережевого обладнання.

Подальші дослідження будуть пов'язані з доопрацюванням розглянутої системи, а також розробкою інших універсальних інструментів для створення Self-Healing мереж.

Список використаних джерел:

1. Білецький В. С. Методологія наукових досліджень технічних об'єктів та їх оптимізація (Навчальний посібник). НТУ «ХПІ». 2023. С. 18 с.

2. Клименко О. Є. Тенденції розвитку самовідновлювальних мереж. Інформаційні технології та суспільство. 2023. № 5 (11). С. 21–27. URL: https://doi.org/10.32689/maup.it.2023.5.3

3. Системи моніторингу та керування – IT-Solutions, Україна. IT-Solutions, Україна. URL: https://it-solutions.ua/servisi/sistemi-monitoringu-ta-keruvannya/

4. AI-on-the-edge-device. URL: https://it-solutions.ua/servisi/sistemi-monitoringu-ta-keruvannya/

5. Al-Oqily I., Bani-Mohammad S., Subaih B., Alshaer J.J. A survey for self-healing architectures and algorithms. *Proc. of the International Multi-Conference on Systems, Signals Devices*. 2012. P. 1–5.

6. Chacon S., Straub B. Pro Git (Second Edition). *Apress open.* 2014. P. 101–122.

7. D. Ghosh Self-healing systems – survey and synthesis. *Decision Support Systems*. 2007. Vol. 42, no. 4. P. 2164–2185.

8. Empson S., Roth H. CCNP ROUTE Command Guide: Implementing Path Control. Cisco Press. 2010. P. 199–208.

9. Manage Kubernetes secrets with SOPS. URL: https://fluxcd.io/flux/guides/mozilla-sops/

10. Ochoa-Aday L., Cervelló-Pastor C., Fernández-Fernández A. Self-healing and SDN: bridging the gap. *Digital Communications and Networks*. 2020. Vol. 6, no. 3. P. 354–368.