УДК 004.72
DOI https://doi.org/10.32689/maup.it.2023.5.3

Олексій КЛИМЕНКО
аспірант кафедри комп'ютерних систем, мереж та кібербезпеки, факультету інформаційних технологій, Національного Університету Біоресурсів та Природокористування України, вул. Героїв Оборони, 16А, Київ, Україна, індекс 03041 (o.klymenko@nubip.edu.ua)

ORCID: 0009-0005-2590-1803

Oleksii KLYMENKO
Postgraduate Student at the Department of Computer Systems, Networks and Cybersecurity, Faculty of Information Technology, National University of Life and Environmental Sciences of Ukraine, 16A, Heroiv Oborony St, Kyiv, Ukraine, postal code 03041 (o.klymenko@nubip.edu.ua)

Бібліографічний опис статті: Клименко, О. (2023). Тенденції розвитку самовідновлювальних мереж. *Інформаційні технології та суспільство*, 5 (11), 21–27. DOI: https://doi.org/10.32689/maup.it.2023.5.3

Bibliographic description of the article: Klymenko, O. (2023). Tendentsii rozvytku samovidnovliuvalnykh merezh [Development tendencies of self-healing networks]. *Informatsiini tekhnolohii ta suspilstvo – Information technology and society*, 5 (11), 21–27. DOI: https://doi.org/10.32689/maup.it.2023.5.3

ТЕНДЕНЦІЇ РОЗВИТКУ САМОВІДНОВЛЮВАЛЬНИХ МЕРЕЖ

Анотація. *У статті розглянуто приклади самовідновлення працездатності мережі під час збоїв, описано етапи розвитку Self-Healing мереж, визначено основні принципи та технології, що відповідають концепції самовідновлення. Комп'ютерні мережі активно змінюються у масштабах та складності технологій. Проте, попри динамічний розвиток технологій концептуальний підхід у вирішенні проблем залишається сталим. Основна тенденція – разом з розвитком та автоматизацією мережі повинні з'являтися інструменти для її автоматичного відновлення, тобто самовідновлення (self-healing). В корні автоматизації лежить програмована складова – скрипти, мови програмування, технології, що дозволяють програмувати ті речі, які раніше не були програмованими. Для цього створюються протоколи, нові програми-прошарки, що здатні впливати на сталі процеси, нові технології. Можливе також використання пропрієтарних механізмів від розробників обладнання. Комп'ютерні мережі стають програмованими з єдиними центром керування. Щоб підтримувати систему моніторингу в актуальному стані та мати практичний зиск, потрібно розуміти тенденції розвитку мереж. Self-Healing мережа здатна здійснювати розширений моніторинг і виконувати коригувальні задачі, які зазвичай потребують втручання людини. Це дає можливість зменшити витрати на IT-персонал і більш стратегічно розподіляти людські ресурси, скорочуючи кількість годин, витрачених на реактивну роботу, та зосереджуватися більше на проактивній діяльності. Хоча концепція самовідновлення почалася понад 10 років тому, її актуальність набирає обертів на фоні активного переходу до SDN-мереж. Автоматизація моніторингу та підтримки працездатності мережі є невід'ємною складовою подальшого розвитку комп'ютерних мереж. Self-Healing мережа, що базується на різних інструментах програмування, позбавлена залежності від конкретного розробника обладнання та є більш універсальною.*

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

DEVELOPMENT TENDENCIES OF SELF-HEALING NETWORKS

Abstract. *The article considers examples of self-healing network performance during failures, describes the stages of development of self-healing networks, defines the main principles and technologies corresponding to the concept of self-healing. Computer networks are actively changing in scale and complexity of technologies. However, despite the dynamic development of technologies, the conceptual approach to solving problems remains stable. The main trend is that along with the development and automation of the network, tools for its automatic recovery, i.e. self-healing, should appear. The root of automation is the programmable component – scripts, programming languages, technologies that allow programming those things that were not programmable before. For this, protocols, new layer programs capable of influencing stable processes, and new technologies are being created. It is also possible to use proprietary mechanisms from equipment developers. Computer networks are becoming programmable with a single control center. To keep the monitoring system up-to-date and to have a practical benefit, you need to understand the development trends of the networks. A self-healing network is capable of advanced monitoring and corrective tasks that would normally require human intervention. This enables to reduce IT staff costs and allocate human resources more strategically, reducing the number of hours spent on reactive work and focusing more on proactive activities. Although the concept of self-healing began more than 10 years ago, its relevance is gaining momentum against the background of the active transition to SDN networks. Automation of monitoring and maintenance of network performance is an integral component of the further development of computer networks. A self-healing network, which is based on various programming tools, is free from dependence on a specific hardware developer and is universal.*

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

Вступ. У сучасних мережах завдання самовідновлення сервісів виходить на новий рівень. У великих сервіс-провайдерів практикується підхід, що сервіс, який перестав працювати, треба швидко вивести з експлуатації та підняти новий сервіс, замість того, щоб витрачати час на пошук причини. Це означає, що потрібно налаштувати системи моніторингу, які протягом секунд виявлять найменші відхилення від норми. Проте, звичних метрик замало, таких як завантаження інтерфейсу або доступності вузла. Недостатньо і ручного стеження чергового інженеру за ними. Для багатьох речей має бути Self-Healing – самовідновлення роботи у разі виникнення проблеми. Відслідковувати потрібно не лише окремі пристрої, а й здоров'я мережі. Це невід'ємна частина автоматизації.

Постановка проблеми. В сучасній розгорнутій динамічній мережі стає все важче, або майже неможливо, вчасно помічати проблеми та реагувати на них у ручному режимі. Процес автоматизації повинен стосуватися не лише Control Plane чи Data Plane, що присутнє у SDN-мережах, але й моніторингу.

Self-Healing (самовідновлювальна) мережа необхідна для мінімізації людських зусиль і витрат, пов'язаних із визначенням причин збою в складних системах. Потрібен час, щоб побачити мережі, які стануть достатньо інтелектуальними для контролю, ідентифікації та виправлення збоїв під час їх виявлення, але дослідження Self-Healing мереж тривають [1].

Аналіз досліджень і публікацій. Визначення Self-Healing мереж добре розкрите в дослідженні [15]. Цей термін означає здатність мережі відновлювати свою роботу незалежно та без зовнішнього втручання у разі будь-якого збою. Кожна система з властивостями самовідновлення має здатність виявляти, діагностувати та реагувати на збої. Основною метою інтеграції функцій самовідновлення в будь-яку роботу мережі є підвищення її надійності та зручності обслуговування. Ці атрибути якості традиційно підвищуються в системах самовідновлення [10; 11].

SDN-мережі останні роки досить активно розвиваються. Технологія постійно досліджується. Self-Healing мережі мають значно менше досліджень, хоча використовуються як у SDN, так і в класичних мережах.

Одне з таких досліджень розкрито у статті [14]. Інженери запропонували модель, в якій мережі Байєса використовуються для діагностування причини аварії, що сталася. Діагностичний блок використовує алгоритм байєсівських мереж, який включає спостереження, що надходять з повідомлень про помилки від NMS (Network Management System) і SM (Service Manage), і додає їх як докази для діагностики першопричини. Цей блок використовує інформацію про топологію мережі, надану контролером SDN, для побудови графа мережі Байєса. Також система отримує статус послуги від SM. Мережі Байєса – це модельний алгоритм, який моделює складні залежності мережі в умовах невизначеності. Алгоритм на основі байєсівської мережі вводиться у SDN, щоб дозволити цій архітектурі виявляти збої в площині додатків, площині керування та площині даних [14].

В задачах моніторингу та самовідновлення мереж також широко використовується протокол OpenFlow [3; 7; 12; 15]. Однак, OpenFlow доречно використовувати тільки в SDN-мережах. Крім того, попри проведені дослідження OpenFlow поки не став універсальним інструментом.

Метою статті є висвітлення тенденцій розвитку Self-Healing мереж, визначення основних принципів та технологій, що відповідають концепції самовідновлення.

Виклад основного матеріалу.

Моніторинг можна розділити на два основні типи:

1) Проактивний – дає можливість побудувати плани розвитку інфраструктури та оцінити результативність сценаріїв по внесенню змін до інфраструктури. Також є можливість прогнозувати та запобігти майбутнім системним недолікам та попередити несправності заздалегідь відносно зібраних історичних даних, та провести аналіз поведінки інфраструктури [13].

2) 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 [13].

При цьому поняття Self-Healing доцільне в кожному з визначних типів. Якщо проактивний моніторинг сигналізує про потенційне "слабке" місце в мережі, потрібно вжити автоматизовані превентивні заходи для запобігання аварії. Якщо реактивний моніторинг зафіксував аварію, Self-Healing мережа має автоматично побудувати альтернативні шляхи трафіку, в той час як інженер займається вирішенням проблеми, що сталася.

Управління складними мережами частіше за все дороге та вимагає великої кількості IT-персоналу. Крім того, час, потрібний для виявлення першопричин і вжиття заходів для виправлення проблем, може призвести до більшої втрати прибутку для компаній [2]. Самовідновлення призведе до скорочення або майже повного усунення процесу вирішення проблеми.

Етапи розвитку самовідновлювальних мереж. Одним з базових принципів самовідновлювальності роботи мережі є метод резервації. Якщо на border-маршрутизаторі перестає працювати основний Інтернет-канал, потрібно це виявити та переключити на резервний канал. Якщо зламався пристрій, потрібно пустити трафік альтернативними шляхами через інші пристрої. Кількість резервних одиниць (канал, пристрій тощо) залежить від потрібного рівня відмовостійкості.

Приклад 1. Один з поширених прикладів відмовостійкої мережі початку XXI сторіччя [4] складається з двох маршрутизаторів, які мають окреме підключення до ISP (Internet Service Provider) та мають між собою декілька з'єднань (рис. 1). У таблиці 1 наведено конфігурацію маршрутизаторів у класичному прикладі резервації каналів.

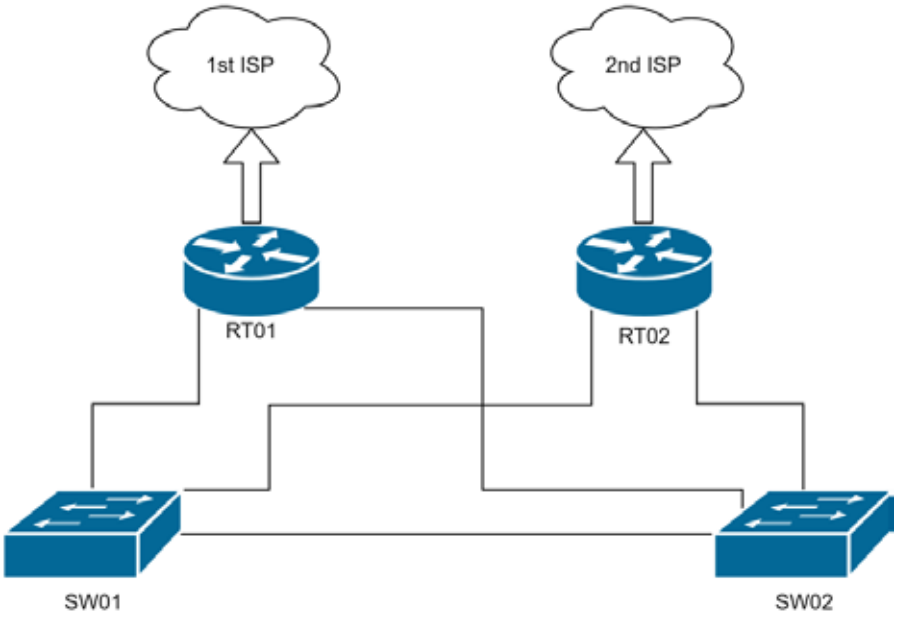

Рис. 1. Класичний приклад резервації каналів

В даній конфігурації потрібно звернути увагу на стек 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 дві секунди 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.

Приклад 2. Наступний підхід стосується мережі ДЦ (дата-центр). В подібній мережі використовується велика кількість серверів та мережевого обладнання [5]. З'єднання будуються вичерпним чином, тобто є зарезервованими. У разі виникнення втрат чи перевантаження каналів на якомусь спайні (комутаторі), трафік потрібно м'яко (тобто без розриву з'єднання) перенаправити через інші маршрути. На рисунку 2 проблема виникає на верхньому спайні, трафік перенаправлено через нижній спайн [6].

Для вирішення подібної проблеми в складній мережі ДЦ використовується складний стек технологій MPTCP (MultiPath Transmission Control Protocol), Flow Label IPv6, eBPF (extended Berkeley Packet Filter). Поле заголовку IPv6 Flow Label з'являється в IPv6 (його немає у IPv4) і воно займає 20 біт. eBPF як невелику програму на C можна вставити в різних місцях виконання стека ядра та TCP-стека. За допомогою eBPF можна динамічно змінювати різноманітні налаштування TCP, в тому числі знизити таймери RTO та SYN-RTO – вплинути на час реакції на подію. Програмування ядра – низькорівнева задача, що вимагає високого рівня компетенцій та навичок.

Між конфігурацією у наведених прикладах різниця у понад 10 років, при цьому: топологія мережі – різна, масштаби – різні, складність – кардинально різна, основна задача в обох випадках – одна: виявити збій та перенаправити трафік альтернативним маршрутом. І тільки наступним кроком настає етап для з'ясування причини збою.

Таблиця 1

Конфігурація маршрутизаторів

Перший маршрутизатор	Другий маршрутизатор
track 1 ip sla 1 reachability delay down 5 up 5 ! interface vlan100 description -=LAN=- ip address 10.1.100.253 255.255.255.0 ip nat inside standby 100 ip 10.1.100.254 standby 100 priority 150 standby 100 track 101 decrement 60 ! ! interface GigabitEthernet0/0/1 description -=1st ISP=- ip address 1.1.1.2 255.255.255.0 ip nat outside ! ip route 0.0.0.0 0.0.0.0 1.1.1.1 ip route 10.0.0.0 255.0.0.0 10.1.100.252 250 ! ip sla 1 icmp-echo 8.8.8.8 source-interface GigabitEthernet0/0/1 threshold 1500 timeout 2000 frequency 3 ip sla schedule 1 life forever start-time now ! event manager applet ISP1-UP event track 1 state up maxrun 40 action 001 wait 30 action 002 cli command "enable" action 003 cli command "clear ip nat trans *" action 004 syslog msg "EEM cleared nat" action 005 cli command "end" action 006 cli command "exit" event manager applet ISP1-DOWN event track 1 state down maxrun 40 action 001 wait 30 action 002 cli command "enable" action 003 cli command "clear ip nat trans *" action 004 syslog msg "EEM cleared nat trans" action 005 cli command "end" action 006 cli command "exit"	track 2 ip sla 2 reachability delay down 5 up 5 ! interface vlan100 description -=LAN=- ip address 10.1.100.252 255.255.255.0 ip nat inside standby 100 ip 10.1.100.254 standby 100 priority 120 standby 100 preempt standby 100 track 12 decrement 60 ! interface GigabitEthernet0/0/1 description -=2nd ISP=- ip address 2.2.2.2 255.255.255.0 ip nat outside ! ip route 0.0.0.0 0.0.0.0 2.2.2.1 ip route 10.0.0.0 255.0.0.0 10.1.100.253 250 ! ip sla 2 icmp-echo 8.8.8.8 source-interface GigabitEthernet0/0/1 threshold 1500 timeout 2000 frequency 3 ip sla schedule 2 life forever start-time now ! event manager applet ISP2-UP event track 2 state up maxrun 40 action 001 wait 30 action 002 cli command "enable" action 003 cli command "clear ip nat trans *" action 004 syslog msg "EEM cleared nat trans" action 005 cli command "end" action 006 cli command "exit" event manager applet ISP2-DOWN event track 2 state down maxrun 40 action 001 wait 30 action 002 cli command "enable" action 003 cli command "clear ip nat trans *" action 004 syslog msg "EEM cleared nat trans" action 005 cli command "end" action 006 cli command "exit"

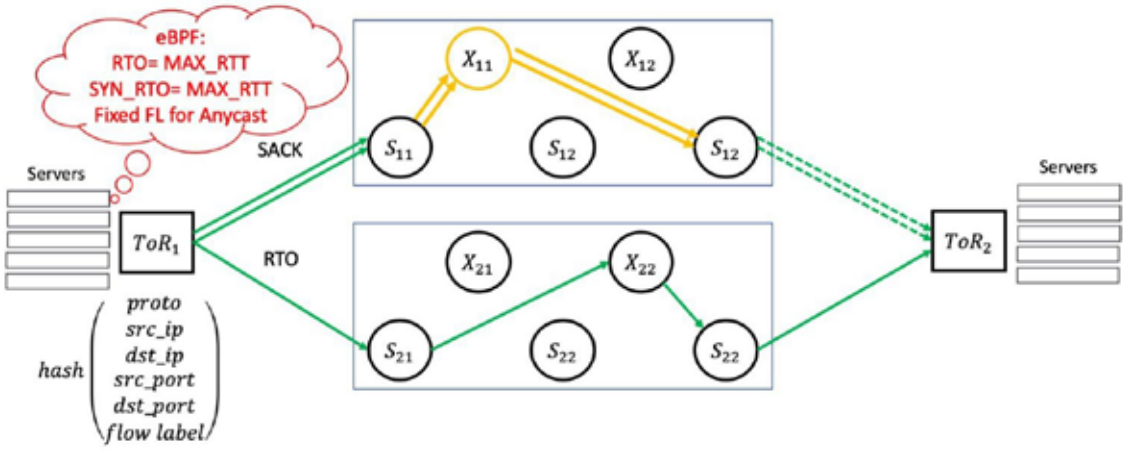

Рис. 2. Перенаправлення трафіку через інші спайни

Звідси можна отримати формулу часу реакції на збій:

$$T = (t_1 - t_0) + (t_2 - t_1) = t_2 - t_0, \quad (1)$$

де t_0 – час збою, t_1 – час виявлення збою, t_2 – час, коли було вжито дії.

На рисунку 3 зображено моніторинг зв'язності між ДЦ [7]. Коли виникає проблема, відповідна секція втрачає зелений колір. Інженер NOC (Network Operation Centre) отримує оповіщення та має відреагувати на інцидент. Для цього йому доведеться проаналізувати додаткові дані, ймовірно, зайти на обладнання, подивитися log-файли.

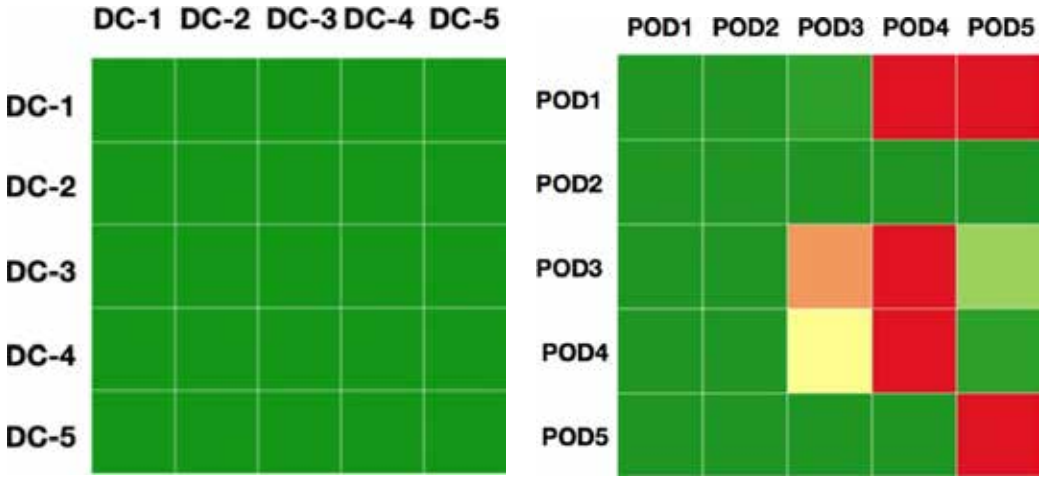

Рис. 3. Зв'язаність між різними ДЦ

До цього моменту було розглянуто приклади, де система чи пристрій автоматично виявляли збій (t_1) та автоматично вживали дії щодо його усунення (t_2). Відповідні таймери можна регулювати в протоколах та впливати таким чином на час реакції на подію. У випадку, коли інцидент обробляє NOC, t_1 залежить від налаштувань моніторингу, на нього також можна впливати; t_2 у свою чергу повністю залежить від роботи інженера та є важко передбачуваним.

Для того, щоб виявити збій, потрібен моніторинг. Мережа кожного окремого підприємства індивідуальна та може виконувати різні специфічні задачі. Доцільність моніторингу того чи іншого параметру мережі визначають інженери компанії. Більше того, деякі проблеми можуть бути подібно до «zero day» – до сьогодні невідомими. Відповідно невідомо, що потрібно відслідковувати у моніторингу. Додавати у моніторинг все не тільки недоцільно, але й неможливо.

Для збільшення прогнозованості часу реакції на подію потрібна автоматизація процесу.

Приклад 3. Нижче наведено фрагмент коду, де скрипт перевіряє коректність введеної команди на обладнанні. Якщо було задано помилкову команду, система виправляє конфігурацію на потрібну. Для реалізації використовується стек NETCONF та Python. Подібні скрипти використовуються з метою вичитки конфігурації після молодших інженерів, виправлення помилок, їх додавання у журнали з метою подальшої роботи над помилками при навчанні інженерів.

```
interface_type = "GigabitEthernet"
interface_id = 1
desired_description = "Tunnel endpoint to cloud provider"

configured+description = get_interface_description(device=DEVICE,
                                        interface_type=interface_type,
                                        interface_id=interface_id)

print(f"Desired interface description: '{desired_description}'\n"
   f"Configured interface description: '{configured_description}'")
```

```
if configured_description != desired_description:
   print("Desired does NOT match configured. Updating…")
   configure_interface_description(device=DEVICE,
                           interface_type=interface_type,
                           interface_id=interface_id,
                           interface_description=desired_description)
else:
   print("Desired description matches configured!")
```

Іншим прикладом може бути ситуація, коли мережу автоматизовано настільки ретельно, що вона не дозволяє інженеру руками вносити зміни на обладнані. Натомість попереджає, що усі зміни в мережі мають відбуватися через оркестратор, тобто через центральний стек серверів, що керують програмованою мережею.

Приклад 4. Self-Healing мережа має прагнути до автоматизації максимальної кількості етапів. У прикладах 1-2 відбувається автоматизоване переключення на резервні канали чи обладнання, але нічого не відбувається для усунення причини проблеми. Якщо комутатор вийшов з ладу, на сьогодні ще немає автоматизованих інструментів замінити його фізично. Проте, у разі падіння Інтернет-каналу процес його відновлення можна частково автоматизувати. Наприклад:

1. Система моніторингу фіксує збій у роботі каналу.
2. Запускає траблшутінг-скрипт на обладнанні.
3. Автоматично аналізує результати.
4. У разі наявності характерних «червоних прапорців» відправляє повідомлення поштою до ISP.
5. Якщо проблема була на стороні ISP, теоретично канал можуть відновити навіть віддалено. Або буде подальша діагностика, а даний етап отримує характер напівавтоматизованого.

Подібний сценарій, окрім мови програмування для написання скрипту, додатково потребує щонайменше протокол доставки команд на обладнання.

Загалом дослідження в напрямку повністю самовідновлювальної мережі тривають.

Частково програмування можуть замінити певні технології від розробників обладнання. Наприклад, Cisco NAE (Network Assurance Engine) – це комплексне рішення для мереж ДЦ, створене на основі запатентованої технології мережевої перевірки Cisco. Проте працювати воно буде лише на обладнанні від Cisco і лише на конкретних моделях з конкретною версією ПЗ (програмне забезпечення) [8; 9].

Висновки. У всіх прикладах неминуче присутні інструменти програмування. EEM – це функціонал, вбудований у Cisco IOS, який дозволяє створювати сценарії для автоматизації роботи пристроїв. eBFP – це підсистема ядра, що дає можливість писати невеликі програми, які будуть запущені ядром у відповідь на події. Python – це мова програмування високого рівня загального призначення. На місці Python можуть бути інші мови.

Отже, Self-Healing мережа обов'язково має програмовану складову, за допомогою якої і відбувається самовідновлення. Мережі майбутнього мають бути програмованими з централізованим керуванням.

Абсолютно все в мережі відслідковувати неможливо. Варто концентрувати увагу на дійсно потрібних метриках та за можливості намагатися автоматизувати в якості реакції на оповіщення процес відновлення мережі.

Сама мережа є динамічною, автоматизація – живий та постійний процес. Оскільки кожна мережа є індивідуальною, процес автоматизації також не може бути шаблонним. Необхідно володіти навичками програмування для автоматизації та гнучко поєднувати Open Source продукти з пропрієтарними продуктами від розробників обладнання.

Альтернативою програмуванню можуть бути пропрієтарні технології від розробників. Проте, при роботі з подібними інструменти інженер працює з певними абстракціями, за якими знову ж таки ховається код від виробника – програмована складова. Подібні технології вимагають повної підтримки обладнанням: тільки конкретний розробник обладнання, тільки на конкретних прошивках для обладнання. Оскільки складно побудувати мережу повністю на обладнанні лише одного розробника, найчастіше дані технології можуть автоматизувати тільки частину мережі, або не працювати взагалі.

Подальші дослідження будуть пов'язані зі створенням універсальних інструментів для самовідновлення мереж як програмно-визначених, так і традиційних.

Список використаних джерел:

1. What does the self-healing network of tomorrow look like? URL: https://irisns.com/2014/11/21/self-healing-network-tomorrow-look-like/

2. Збій Facebook: чому він стався і тривав так довго. BBC News Україна. URL: https://www.bbc.com/ukrainian/features-58795279

3. Thorat P., Jeon S., Choo H. Enhanced local detouring mechanisms for rapid and lightweight failure recovery in OpenFlow networks. *Computer Communications.* 2017. Vol. 108. P. 78-93.

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

5. Clos Networks: What's Old Is New Again. Network World. URL: https://www.networkworld.com/article/2226122/clos-networks-what-s-old-is-new-again.html

6. Self healing Network The Magic of Flow Label – IETF Datatracker. URL: https://datatracker.ietf.org/meeting/111/materials/slides-111-rtgwg-sessb-3-selfhealing-network-01

7. Adrichem N.L. M.v., Asten B.J.v., Kuipers F.A. Fast recovery in software-defined networks. *Proc. Of the 3rd European Workshop on, Software-Defined Networks*. 2014. P. 61-66.

8. Cisco Network Assurance Engine – Cisco Network Assurance Engine At-a-Glance. Cisco. URL: https://www.cisco.com/c/en/us/products/collateral/data-center-analytics/network-assurance-engine/at-a-glance-c45-740230.html

9. Building self-healing networks with Cisco Network Assurance Innovations and Service Now ITSM. Cisco Blogs. URL: https://blogs.cisco.com/datacenter/building-self-healing-networks-with-cisco-network-assurance-innovations-and-servicenow-itsm

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

11. 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.

12. Cascone C., Pollini L., Sanvito D., Capone A. Traffic management applications for stateful SDN data plane. *Proc. of the Fourth European Workshop on Software-Defined Networks*. 2015. P. 85-90.

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

14. Sanchez J., Ben Yahia I.G., Crespi N. Self-Healing Mechanisms for Software Defined Networks. *8th International Conference on Autonomous Infrastructure, Management and Security.* 2014.

15. 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.

References:

1. What does the self-healing network of tomorrow look like? (n.d.). Retrieved from: https://irisns.com/2014/11/21/self-healing-network-tomorrow-look-like/

2. Zbiy Facebook: chomu vin stavsya i tryvav tak dovho [The Facebook crash: why it happened and lasted so long]. (2021). BBC News Ukraine. Retrieved from: https://www.bbc.com/ukrainian/features-58795279 [in Ukrainian].

3. Thorat P., Jeon S., Choo H. (2017). Enhanced local detouring mechanisms for rapid and lightweight failure recovery in OpenFlow networks. *Computer Communications.* Vol. 108. P. 78-93.

4. Empson S., Roth H. (2010). CCNP ROUTE Command Guide: Implementing Path Control. *Cisco Press.* P. 199-208.

5. Clos Networks: What's Old Is New Again. Network World. (2014). Retrieved from: https://www.networkworld.com/article/2226122/clos-networks-what-s-old-is-new-again.html

6. Self healing Network The Magic of Flow Label – IETF Datatracker. (n.d.). Retrieved from: https://datatracker.ietf.org/meeting/111/materials/slides-111-rtgwg-sessb-3-selfhealing-network-01

7. Adrichem N.L. M.v., Asten B.J.v., Kuipers F.A. (2014). Fast recovery in software-defined networks. *Proc. Of the 3rd European Workshop on, Software-Defined Networks*. P. 61-66.

8. Cisco Network Assurance Engine – Cisco Network Assurance Engine At-a-Glance. Cisco. (2021). Retrieved from: https://www.cisco.com/c/en/us/products/collateral/data-center-analytics/network-assurance-engine/at-a-glance-c45-740230.html

9. Building self-healing networks with Cisco Network Assurance Innovations and Service Now ITSM. Cisco Blogs. (2021). Retrieved from: https://blogs.cisco.com/datacenter/building-self-healing-networks-with-cisco-network-assurance-innovations-and-servicenow-itsm

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

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

12. Cascone C., Pollini L., Sanvito D., Capone A. (2015). Traffic management applications for stateful SDN data plane. *Proc. of the Fourth European Workshop on Software-Defined Networks*. P. 85-90.

13. Systemy monitorynhu ta keruvannya [Systems of monitoring and management] – IT-Solutions, Ukraine. IT-Solutions, Ukraine. (n.d.). Retrieved from: https://it-solutions.ua/servisi/sistemi-monitoringu-ta-keruvannya/ [in Ukrainian].

14. Sanchez J., Ben Yahia I.G., Crespi N. 2014. Self-Healing Mechanisms for Software Defined Networks. *8th International Conference on Autonomous Infrastructure, Management and Security*.

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