УДК 621.391

Гринкевич Г.О., к.т.н.; **Макаренко А.О.,** к.т.н.; **Жебка В.В.,** к.т.н.;
Куклов В.М., аспірант; **Підручний А.І.,** аспірант

АНАЛІЗ ПРАЦЕЗДАТНОСТІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

Grynkevych G.O., Makarenko A.O., Zhebka V.V., Kuklov V.M., Pidruchniy A.I. Analysis serviceability program-configured networks.

The paper presents the concept of Software-Defined-Networking, and the history of programmable networks and associated review of the latest standards. One of them, the OpenFlow protocol, played an important role in promoting the concept of SDN, significantly influenced research and industry. Consider in more detail the basic principles of OpenFlow. Proposed expansion OpenFlow specification for deployment of decentralized monitoring network, including packet generation and processing on network devices. Technology can restore protection OpenFlow network within 50 ms. For this purpose any action from the network controller is not required, because the switch can immediately respond to setbacks. The technique of restoration controller must interact with network devices that takes longer and makes the method less convenient for large networks with many made components.

Keywords: information network, software-configured network, information security, Linux, Ethernet, OpenFlow.

Гринкевич Г.О., Макаренко А.О., Жебка В.В., Куклов В.М., Підручний А.І. Аналіз працездатності програмно-конфігурованих мереж.

У статті представлено концепцію Software-Defined-Networking, а також історію програмованих мереж і огляд пов'язаних з ними новітніх стандартів. Один з них, протокол OpenFlow, відіграв важливу роль у просуванні концепції SDN, значно вплинув на наукові дослідження і промисловість. Розглянуто більш детально основні принципи OpenFlow.

Ключові слова: інформаційні мережі, програмно-конфігуровані мережі, інформаційна безпека, Linux, Ethernet, OpenFlow.

Гринкевич А.А., Макаренко А.А., Жебка В.В., Куклов В.М., Пидручный А.И. Анализ работоспособности программно-конфигурируемых сетей.

В статье представлена концепция Software-Defined-Networking, а также история программируемых сетей и обзор связанных с ними новых стандартов. Один из них, протокол OpenFlow, сыграл важную роль в продвижении концепции SDN, значительно повлиял на научные исследования и промышленность.

Ключевые слова: информационные сети, программно-конфигурируемые сети, информационная безопасность, Linux, Ethernet, OpenFlow.

Вступ

Постановка задачі. В літературі і в промисловості, концепція програмованих мереж або Software-Defined-Networking (SDN) все ще використовується розробниками, які мають власні ідеї стосовно її оновлення. Фонд Open Network [1] – організація для користувачів з метою сприяння розвитку SDN, описує його як "Фізичне розділення управління мережами в площині від переадресації і до площини управління, що управляє декількома пристроями".

Термін "програмованість" відноситься до гнучкості в площині управління, яка безпосередньо опрацьовується в програмному забезпеченні і включає в себе канал керування для обміну даними з мережевих пристроїв. Інтерфейс (API) дозволяє додаткам взаємодіяти з площиною управління для отримання доступу і зміни мережевих послуг. Наслідком цього виступає новий принцип конструкції, що підвищує гнучкість в управлінні мережею. За аналогією, цю еволюцію порівнюють з переходом від телефонів з обмеженою гнучкістю до смартфонів з операційною системою і більш широким функціоналом. Концепція традиційних комп'ютерних мереж ґрунтувалась на надмережевих пристроях, які містять як дані, так і площини управління. Для вирізнення від парадигми SDN, назвемо їх непрограмованими мережами.

Передумови. Парадигма програмованих мереж попередньої розробки, активно використовувалася в середині 1990-х років. Зусилля дослідників, зосереджені навколо програмованих мереж, були розділені на два напрями. [2]. Перша школа, так звана активна мережа (AN), складалась з великої групи проектів і була заснована DARPA. Вона характеризується гнучким і динамічним підходом в напрямку провадження послуг в мережі. Наприклад, один пакет може бути використаний як для для зміни мережевих послуг, так і для забезпечення максимуму гнучкості у конфігурації мережі.

На противагу йому, OpenSignalling (OpenSig) запропонували ідею створення стандартизованих інтерфейсів для зв'язку з мережевими пристроями, що була реалізована за допомогою IEEE P1520 [3]. Маршрутизатор і комутатори мають відкритий доступ використання інтерфейсів програмування (API), щоб надати можливість розробляти додаткове програмне забезпечення для нього. Незважаючи на дослідження в співтоваристві OpenSig, центрування навколо програмованих мереж, промисловість не адаптувалася до ідеї програмованих мереж зі стандартизованим API. Це призвело до патентованих протоколів із закритою складовою пристрою, що характеризує сучасні мережі.

1. Архітектури і інтерфейси

Різноманіття архітектур промисловості і наукових співтовариств розвивалися паралельно з моменту створення SDN.

Галузі в основному засновані на власному розумінні SDN, хоча деякі з них були стандартизовані або, принаймні, обговорюються в робочих групах Internet Engineering Task Force (IETF2).

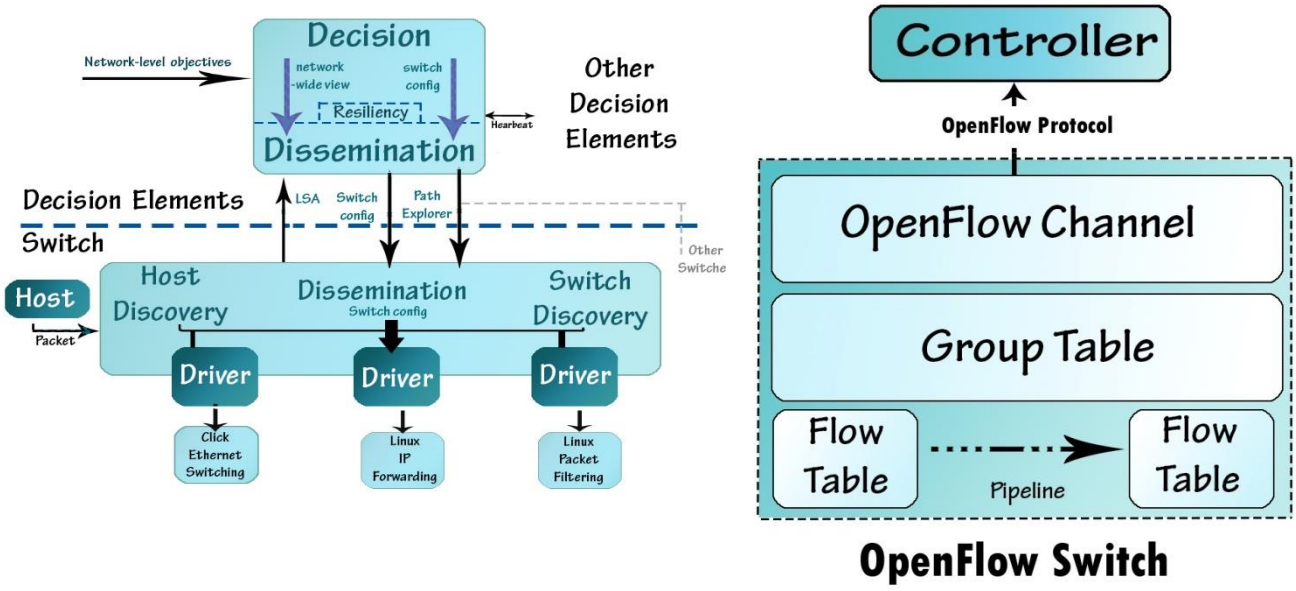
Для того, щоб прискорити інновації в вендора нейтральним чином і визначити стандарти для SDN, Фонд Linux встановив (ODL) 3 лабораторних проекта OpenDaylight. Ця відкрита структура в даний час підтримується 29-ма провідними членами промисловості, такими як, Cisco, IBM, Juniper і NEC. Проект заснований на існуючих інтерфейсах і протоколах (наприклад, OpenFlow і NETCONF [4]), в даний час може розглядатися як найбільш перспективний крок на шляху розвитку SDN. Нижче вказані підходи, що виникли навколо SDN.

У той час як в OpenFlow більшість функціональних можливостей управління є централізованими, Tessteract розміщує інтелект в комутаторі.

Платформа маршрутизації управління (RCP). Робота RCP [5] виконується за допомогою AT&TLabs для того, щоб подолати обмеження вибору шляху до Інтернету. Кожен RCP присвоюється автономній системі (AS) з протоколом Inter-AS та інформацією про маршрутизацію між ними. В межах RCP, BorderGatewayProtocol (BGP) використовується в якості протоколу управління для обміну даними з маршрутизаторами. Вони містять площину даних на основі таблиць IP-експедиторських, які забезпечують лише невелику частину списку полів в OpenFlow.

Tesseract. Конструкція Tesseract [6] заснована на архітектурі 4D, що приймає рішення про поширення, виявлення і дані площини, як вказано на рис. 1 (а). Комутатори вмикають виявлення тегу задля знахождения хостів, комутаторів та для передачі пакетів, грунтуючись на площині даних. Тессеракт дозволяє здійснювати прямий контроль комп'ютерної мережі шляхом відділення логіки прийняття рішень (наприклад, маршрут обчислення) з нижчих протоколів. Цей підхід може також використовуватися для розгортання централізованої політики управління в мережі. У порівнянні з архітектурою OpenFlow, показаної на рис. 1 (б), мережеві пристрої містять тільки потік таблиць і спрощені вимоги до обчислювальної техніки.

Експедиція і управління елементом розподілу (Forces). Мета робочої групи (RFC 5810 [5]) полягає в розробці протоколу для Software-DrivenNetworking. Це вказує на деяку схожість з протоколом OpenFlow, окрім того, що площина даних – структурована. У той час як OpenFlow заснований на потоці таблиць, Forces використовує LFBs (логічні функціональні блоки), щоб встановити канали передачі даних і створення пакета потоків.



(а) Tesseract (б) OpenFlow

Рис. 1. Архітектура Tesseract і OpenFlow

ONF і OpenFlow. Поява SDN в значній мірі сприяла розвитку OpenFlow [1], відкритого протоколу, який використовується для обміну даними між централізованим мережним контролером і мережевими пристроями. Вона була ініційованою Мартіном Касадо в Етан [2], SANE [3] і CleanSlate проекта в Стенфордському університеті (табл. 1). В 2008 році OpenFlow був успішно розгорнутий в їх мережі з використанням обладнання комутаторів, що містили тільки площину даних, управління з котрого перемістилось на мережевий контролер (рис. 2). Специфіки OpenFlow визначені і розроблені ONF.

Таблиця 1. Архітектури і інтерфейси для SDN

Архітектура	Інтерфейси	Централізація	Стандартизації	Підтримка промисловості
RCP	BGP, Inter-AS	середня	-	-
Тессеракт	множинний	середня	-	-
-	ForCES	висока	RFC 5810	-
ODL	множинний	середня	-	Всі основні постачальники
ONF	OpenFlow	висока	-	Всі основні постачальники
PCE	pCEP	низький	RFC 4655	-
-	I2RS	низький	IETF Проект	Всі основні постачальники
ONE	onePK	низький	патентований	Cisco

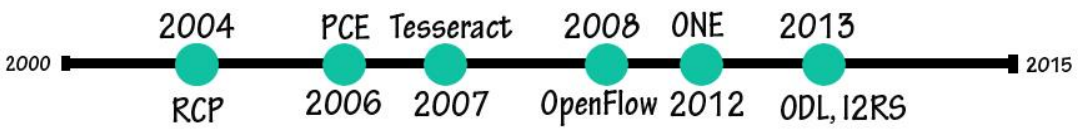


Рис. 2. Хронологія історичного розвитку SDN

Елемент шляху обчислення (PCE) PCE є програмним компонентом, який використовується для розрахунку шляху через ярлик багатопротокольної комутації по мітках

(MPLS) мережі. Такий підхід призводить до виникнення мережевих пристроїв, в котрих видаляється з площини управління тільки компонент розрахунку управління. У той час як PCE надає менше функціональних можливостей і гнучкості ніж OpenFlow, її розгортання в глобальних мережах (WAN) менше, ніж для операторів, так як тільки крайові маршрутизатори повинні бути модифіковані [7]. PCE був стандартизований IETF, наприклад, в RFC 4655 [8].

Інтерфейс до системи маршрутизації (I2RS) I2RS являє собою робочу групу в IETF, яка має підтримку з боку ряду постачальників. Вона зберігає дані і керуючі компоненти в мережевих пристроях. На противагу пропонує інтерфейс, який дозволяє налаштувати зв'язок з площинами управління, які можуть бути розташовані за межами локальної мережі. Це дозволяє взаємодіяти з протоколами маршрутизації від зовнішнього програмного забезпечення керованого екземпляру.

Open Network Environment (ONE). Управління SDN вимагає декількох API-інтерфейсів, наприклад, одного каналу доступу до мережевих пристроїв в той час як другий буде керувати обміном даних між мережевими службами і зовнішніми додатками. ONE компанії Cisco [9] це архітектура інтерфейсу One Platform Kit (OnePK), що були об`єднані в єдине рішення. Це дозволяє розгортання мережевих додатків не тільки на виділеному контролері мережі, як запропоновано в OpenFlow, а й на самих мережевих пристроях. Також це виключає використання товарного обладнання, але дозволяє більш гнучке розбиття функціональності між даними і площинами управління.

Крім вказаних архітектур загальний дизайн SDN все ще розвивається. Наприклад, Fabric (тканинний) [10] підхід поєднує в собі мережі програмованості з MPLS з метою підвищення SDN. Пакети, які входять або виходять з мережі, обробляються за допомогою вхідного/вихідного граничного комутатора, який управляється контролером. Він може надавати мережеві послуги, такі як безпека, фільтрація і ізоляції. Завдання пересилання пакетів по мережі здійснюється через "мережеві ділянки". Такий поділ дозволяє їм і самому перемикачу еволюціонувати окремо, без впливу один на одного.

Термін, який тісно пов'язаний з SDN – є віртуалізацією мережі. Вона пропонує стикування обладнання в програмному забезпеченні, щоб забезпечити певні завдання, наприклад, для визначення кількості віртуальних комутаторів. Вони можуть бути реалізовані, наприклад, з відкритим vSwitch, який заснований на використанні відкритого програмного забезпечення і широко використовується в операційних сценаріях. Поняття віртуалізація мережі також можуть бути використані в якості основи для розробки рішень SDN.

OpenFlow на основі SDN. Ця стаття фокусується на OpenFlow – основі парадигми SDN, так як більшість поточних досліджень і програмного забезпечення з відкритим вихідним кодом зосереджена саме навколо нього. В результаті, він є все найбільш розгорнутим у виробничому середовищі. Наприклад, Google використовує концепцію OpenFlow на основі SDN для їх внутрішньої магістральної мережі під назвою G-Scale [8], яка є великою міжнародною мережею. Досвід компанії був дуже багатообіцяючим, що дозволило швидкому розгортанню багатьох функцій і спрощенню управління мережею. У промисловості більшість виробників мережевого устаткування забезпечують мережеві пристрої з підтримкою OpenFlow, такі як Cisco, HewlettPackard [11] і NEC. Архітектура SDN з OpenFlow показанана рис. 3 і складається з трьох рівнів.

Інфраструктура рівня. Цей рівень використовується для пересилки пакетів на основі OpenFlow – підтримуваних мережевих пристроїв. Спеціалізоване обладнання, включаючи пам'ять для зберігання потоку інформації, дозволяє швидку обробку пакетів. Типові мережеві пристрої, які належать до цього шару, – це комутатори і маршрутизатори. Комутатор зазвичай працює на рівні ISO7 лінії зв'язку (L2), в той час як маршрутизатор може також обробляти мережевий рівень (L3) інформації, що дозволяє пересилку пакетів за межі локальних мереж (LAN).

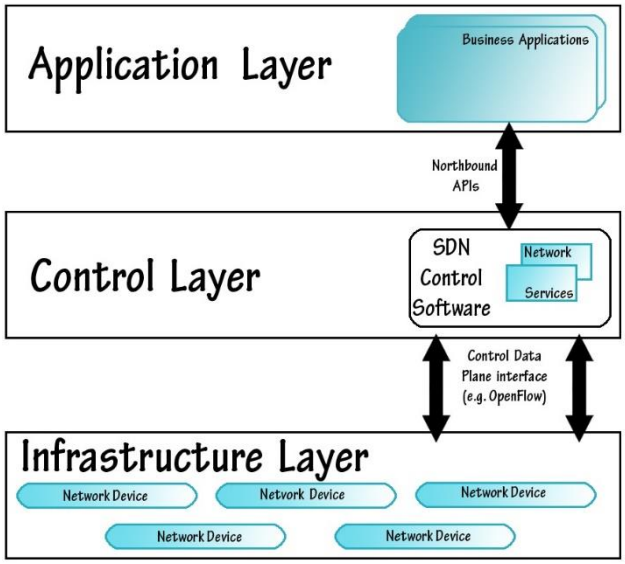


Рис. 3. Конструкція трирівневої програмно-визначеної архітектури

Контроль рівня. Мережевий контролер, який часто використовують в якості операційної системи, включає в себе площину управління і має централізоване подання. Це може бути використано для виконання рішень переадресації, що засновані на врахуванні стану всіх підключених мережевих пристроїв. Контролер здійснює зв'язок з мережевими пристроями через інтерфейс OpenFlow в південному напрямку. Другий тип API, інтерфейс в північному напрямку, використовується для обміну даними між сервісами і бізнес-додатками. Типові приклади мережевих послуг включають в себе маршрутизацію, комутації пакетів (L2) і MPLS. Крім того, кілька контролерів можуть бути розгорнуті з різних причин, наприклад, щоб уникнути єдиної точки відмови або для окремих завдань.

Натхнені першим відкритим вихідним мережевим контролером, NOX [12], написаної в C ++, науково-дослідні інститути і компанії розробили мережеві контролери для широкого кола цілей, з відкритим вихідним кодом спільноти: типові контролери POX [13] (Python), Beacon (Java) і Floodlight (Java). Останній підтримується компанією BigSwitchNetworks, який вважається комерційним.

Прикладний рівень. Даний рівень дозволяє використовувати бізнес-додатки для зміни і надання мережевих послуг клієнтам. Це – вимога визначення API для того, щоб дозволити стороннім розробникам створювати і продавати мережеві додатки до оператора мережі.

Frenetic [14] є мовою програмування мережі, що використовується для визначення політики вищого рівня. Це забезпечує підвищення абстракції втратити зв'язку із мережею для того, щоб приховати обробку пакетів низького рівня від програміста. Проект нещодавно був запропонований Python на основі мови під назвою Pyretic [15]. Це забезпечує створення абстрактної моделі пакету, логічну політику високого рівня і мережевих об'єктів.

Протокол OpenFlow. Інтерес до SDN значно зріс після розробки протоколу OpenFlow, що сьогодні використовується під управлінням ONF [5]. У той час як основна функціональність була повністю реалізована у версії 1.0, більш пізні розробки включають в себе підтримку IPv6 і MPLS.

Мережеві потоки. Протокол OpenFlow заснований на концепції мережевих потоків для моніторингу та управління TRAFFI C всієї мережі. Це поліпшило масштабованість за рахунок збільшення швидкості мережі і в даний час підтримуються потоки з підтримкою пристроїв від всіх основних виробників. Є кілька визначень терміна потоку в літературі. IPFlow робоча група в рамках IETF [16] описує IP потік таким способом: "Потік, що визначається як сукупність IP-пакетів, котрий проходять через точку спостереження в мережі

протягом певного інтервалу часу. Всі пакети, що належать до конкретного потоку, мають ряд загальних властивостей".

Наприклад, зв'язок між двома об'єктами призводить до двох притоків, як показано рис. 5. Типові властивості одного потоку включають в себе джерело і IP-адреси призначення, порту джерела і одержувача, а також тип протоколу (TCP, UDP, ICMP, і т.д.).

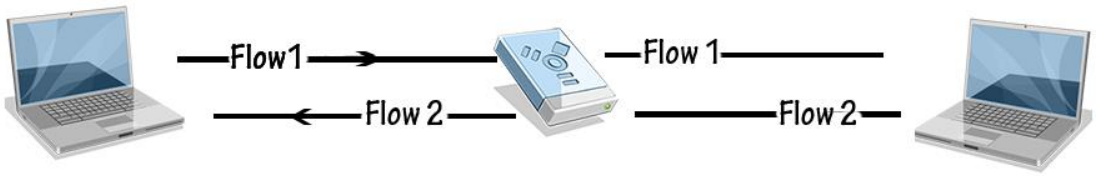


Рис. 4. Flow на основі зв'язку між двома учасниками

Таблиця витрат. Для пересилання пакетів, заснованих на концепції притоків, кожен комутатор OpenFlow має містити одну або кілька потокових таблиць, використовуваних для зберігання попереднього потоку записів/правила. Наприклад, адреса управління доступом до середовища передачі (MAC) може бути використана для ідентифікації вузлів в комутованій мережі. Кожен запис потоку містить наступні компоненти:

Зіставлення полів | пріоритет | Лічильники | інструкції | таймаут | файл Cookie

Для всіх вхідних пакетів, які надходять на розмикач, потік таблиці перегляду виконується шляхом зіставлення заголовка пакета проти Match Fields всіх потоків записів. Кожен OpenFlow на основі комутатора підтримує відповідність полів, зазначених на рис. 6, які можуть бути або містять Specі значення.

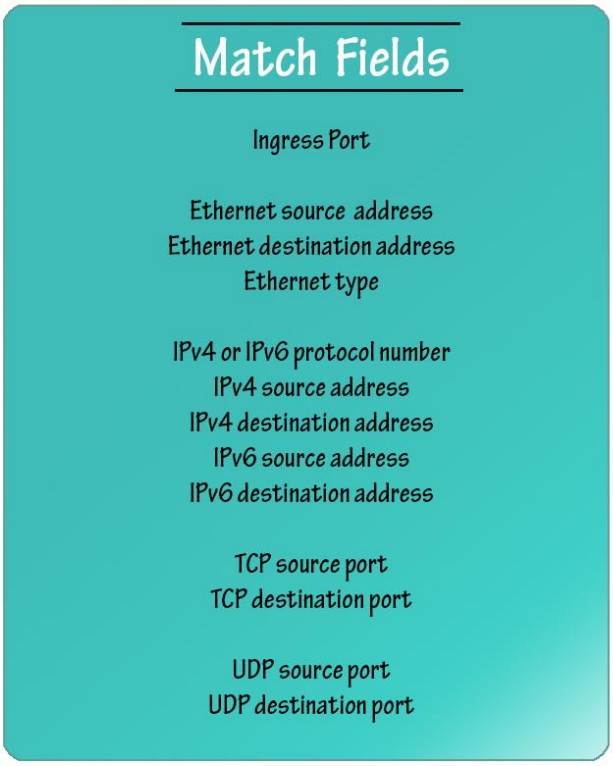


Рис. 5. Match Fields всіх потоків записів

Висновки

В даний час дослідження в області OpenFlow на основі SDN вже допомогли вирішити деякі питання до стадії оперативного розгортання самого проекту. Як вище зазначено, управління несправностями концентрується в основному на виявленні та виправленні переривання зв'язку, які можуть пошкодити не тільки дані, а й зв'язок з контролером. Для того, щоб запропонувати нові схеми виявлення несправностей для SDN, відповідні методи моніторингу для цієї архітектури повинні бути вивчені. Існує декілька підходів до вимірювання та моніторингу мережі, котрі грунтуються на вставці додаткових шарів між пристроями і контролером мережі. Це дозволяє забезпечити перевірку керуючих повідомлень OpenFlow. Наприклад, щоб ідентифікувати трафік перевантаження. Так як додатковий шар відрізняється від оригінальної архітектури OpenFlow і вимагає додаткових апаратних засобів, аналізу потоку записів на мережі контролера. Це може бути досягнуто шляхом розгортання алгоритмів моніторингу, що працюють на основі мережевого контролера. Аналізи потоку можуть бути використані для виявлення характеристик трафіку і побудови повної картини мережі шляхом включення вхідних сигналів від всіх мережевих пристроїв.

Для налагодження програмного забезпечення необхідно забезпечити контроль за пакетами в мережі, хоч формально перевірка може бути використана для визначення правильності потоку. Це необхідно для того, щоб генерувати докладні мережеві знімки, а також інспектувати параметри функції при певних умовах мережі. З точки зору безпеки, потенціал OpenFlow реалізований на нових схемах безпеки не були добре вивченим. Аналізуючи пакети в повідомленні Packet на контролері мережі вони можуть бути використані для ідентифікації і визначення проникнення зв'язку, перш ніж потік налаштується. У той час як для безпеки мережі були досліджені нові вектори атак, централізований контролер мережі і канал управління залишаються частково незахищеними.

Список використаної літератури

1. McKeown N. "OpenFlow: enabling innovation in campus networks," / N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner // SIGCOMM Comput. Commun. Rev, vol. 38, no. 2. -Mar. 2008- pp. 69–74.

2. Дуравкин Е.В. Архитектура SDN. Анализ основных проблем на пути развития [Електронний ресурс] / Е.В. Дуравкин, Е.Б. Ткачева, Иссам Саад// Системи обробки інформації . – 2015. - випуск 3 (128) - Режим доступу: https://www.google.com.ua/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&cad=rja&uact=8&ved=0ahUKEwiM_Mee87DSAhXBvBoKHbr3AhcQFghTMAc&url=http%3A%2F%2Fwww.hups.mil.gov.ua%2Fperiodic-app%2Farticle%2F12126%2Fsoi_2015_3_21.pdf&usg=AFQjCNH6PaaG9b2Jn1SzKgoF2Rh3VlQYG

3. Смелянский Р. Программно-конфигурируемые сети [Электронный ресурс]/ Р. Смелянский // Открытые системы. – 2012. – № 9. – Режим доступа к статье: www.osp.ru/os/2012/09/13039421/

4. О. Р. Лапонина Способы трансформации сетей к SDN-Архитектуре [Електронний ресурс] / О. Р. Лапонина, В.А.Сухомлин // – Режим доступу: http://injoit.org/index.php/j1/article/viewFile/191/143

5. Open Networking Foundation, "SDN Definition," [Електронний ресурс] // Open Networking Foundation. - 2013. – Режим доступу: https://www.opennetworking.org/index.php?option=comcontent&view=article&id=686&Itemid=272&lang=en

6. Wall D.W. "Messages as active agents," / D.W.Wall // in Proceedings of the 9th ACM SIGPLANSIGACT Symposium on Principles of Programming Languages (POPL'82).- New York, NY, USA: ACM. – 1982. - pp. 34–39.

7. Красотин А. А. Программно-конфигурируемые сети как этап эволюции сетевых технологий / А. А. Красотин, И. В.Алексеев // Моделирование и анализ информационных систем. – 2013. – Т. 20. – №. 4. – С. 110-124.

8. Фещенко О. А. Анализ задач, возложенных на программно-конфигурируемые сети // Системи обробки інформації. – 2013. – Випуск 9 (116). – С. 181-183.

9. Роговой В. П. ЦЕНТРЫ ОБРАБОТКИ ДАННЫХ НА БАЗЕ ТЕХНОЛОГИИ SDN Роговой В. П. // Праці УДК 621.3 . – 2013. - №3 . - . С. 16-19 .

10. Smith J. "Active networking: one view of the past, present, and future," Systems, Man, and Cybernetics, Part C: Applications and Reviews. / J. Smith and S. Nettles. // IEEE Transactions on. - Feb. 2004. - vol. 34, no. 1. - pp. 4 –18.

11. Tennenhouse D. "A survey of active network research,"/ D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. // IEEE Communications Magazine. - Jan. 1997. - vol. 35, no. 1. - pp. 80–86.

12. Biswas J. "The IEEE P1520 standards initiative for programmable network interfaces,". / J. Biswas, A. A. Lazar, J. F. Huard, K. Lim, S. Mahjoub, L. F. Pau, M. Suzuki, S. Torstensson, W. Wang, and S. Weinstein. // Comm. Mag. - Oct. 1998. - vol. 36, no. 10. - pp. 64–70.

13. Stadler R. Active Technologies for Network and Service Management. / R. Stadler and B. Stiller. // 10th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'99), ser. Lecture Notes in Computer Science. - Oct. 1999. - vol. 1700. Springer .

14. Brunner M. "Management in telecom environments that are based on active networks," . / M. Brunner and R. Stadler. // J. High Speed Netw. - Dec. 2000. - vol. 9. no. 3,4. - pp. 213–230.

15. Enns R. NETCONF Configuration Protocol (IETF RFC 6241). / R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. - June 2011.

16. Feamster N. "The case for separating routing from routers,". / N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. // in Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA'04). - New York, NY, USA: ACM. – 2004. - pp. 5–12.

Автори статті

Гринкевич Ганна Олександрівна – к.т.н., доцент кафедри телекомунікаційних систем, Державний університет телекомунікацій, Київ, Україна. Тел. +38 067 440 52 75. E-mail: ggrynkevych@i.ua

Макаренко Анатолій Олександрович – кандидат технічних наук, доцент, доцент кафедри Інформаційно-комунікаційних технологій, Державний університет телекомунікацій, Київ, Україна. Тел. +38 097 509 00 33. E-mail: makarenkoa@ukr.net

Жебка Вікторія Вікторівна - к.т.н., доцент кафедри інженерії програмного забезпечення, Державний університет телекомунікацій, Київ, Україна, Тел. +38 098 728 46 13. E-mail: viktoria_zhebka@ukr.net

Куклов Валентин Михайлович – аспірант, асистент кафедри телекомунікаційних систем та мереж, Державний університет телекомунікацій, Київ, Україна. Тел. +38 066 297 14 53. E-mail – vm.kuklov@gmail.com

Підручний Артем Ігорович – аспірант, асистент кафедри Комп'ютерних наук і інформаційних технологій, Державний університет телекомунікацій, Київ, Україна. Тел. +38 093 335 95 42. E-mail: mrtom0311@gmail.com

Authors of the article

Grynkevych Ganna Oleksandrivna – candidate of science (technic), assistant professor, assistant professor of Department of Telecommunication systems, State university of telecommunications, Kyiv, Ukraine. Tel. +38 093 996 26 02. E-mail: ggrynkevych@i.ua.

Makarenko Anatoliy Oleksandrovych – candidate of Science (technic), associate professor, associate professor of Department of Information and communications technology, State University of Telecommunications, Kyiv, Ukraine. Tel. +38 097 509 00 33. E-mail: makarenkoa@ukr.net

Zhebka Viktoria Viktorivna - candidate of science (technic), assistant professor of Department of Application Programming, State university of telecommunications, Kyiv, Ukraine. Tel. +38 098 728 46 13. E-mail: viktoria_zhebka@ukr.net

Kuklov Valentyn Mykhaylovych - postgraduate student, teaching assistant of Department of Telecommunication systems and networks, State university of telecommunications, Kyiv, Ukraine. Tel. +38 066 297 14 53. E-mail: vm.kuklov@gmail.com

Pidruchniy Artem Igorovych - postgraduate student, teaching assistant of Department of Computer Science and information technolog, State university of telecommunications, Kyiv, Ukraine. Tel. +38 093 335 95 42. E-mail: mrtom0311@gmail.com

Дата надходження в редакцію: 08.02.2017 р. Рецензент: д.т.н., проф. В.Ф. Заїка