УДК 621.39
DOI https://doi.org/10.32782/2663-5941/2023.5/11

Романов О.І.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Бурлака Г.Ю.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

УПРАВЛІННЯ МЕРЕЖЕЮ SDN З ВИКОРИСТАННЯМ КОНТРОЛЕРА RYU

На сьогоднішній день з'ясовано що найбільш складні завдання в мережах SDN стоять перед рівнем управління. Вирішення цих завдань покладається на контролер, основним елементом якого є мережева операційна система. У цій статті розглянуто особливості побудови контролера Ryu, протоколи та інтерфейси, функціональний склад елементів. Наведено порівняльну характеристику даного контролерами з іншими та оцінено за певними критеріями, завдяки яким можна отрмати загальну картину позитивних та негативних сторін цього контролера.В цій статті розкрито принципи побудоми мережі mininet з використанням даного контролера.

В цій статті описано що являє собою контролер Ryu а саме відкритий контролер мережі (SDN), призначений для підвищення гнучкості мережі за рахунок посилення управління та адаптації способів обробки трафіку. В загальному, контролер SDN – це мозкова середовище SDN, що передає інформацію про комутатори та маршрутизатори за допомогою південних API, а також про програми та бізнес-логіку за допомогою північних API. Структура Ryu відрізняється від інших рішень тим, що вона надає просту допоміжну інфраструктуру, яку користувачі платформи повинні написати для використання на власний розсуд. Хоча для цього потрібний досвід розробки, це також забезпечує повну гнучкість рішення SDN. Існуючі компоненти можна швидко і легко оновлювати і об'єднувати в існуючі мережі, щоб задовольнити потреби різних програм, що змінюються, використовуючи ці компоненти.

У статті розглянуті компонентні сколадові контролера Ryu його позитивні і негативні сторони в порівнянні з іншими контролерами. Розглянуто особливості побудови контролера Ryu, протоколи та інтерфейси, функціональний склад елементів. Також побудовано мережу SDN з використанням контролера Ryu та емулятора mininet, в цій топології перевырено працездатність контролера та проведено тестування параметрів функціонування мережі, для цього проведено тестування пропускної здатності каналів зв'язку у напрямку від h11 до h9.

Ключові слова: *mininet, Ryu Controller, OpenFlow, ONOS, APP Manager, Ryu Libraries, пропускна здатність.*

Постановка проблеми. Традиційні мережі обмежені рядом вимог до послуг, а також розміром мережі. Ці вимоги стосуються інженерного трафіку, керування потоком, реалізації політики, надійності, і деякі з них включають віртуалізацію. Мережі стають дедалі складнішими, їх важко проектувати та експлуатувати. Програмно визначена мережа (SDN) – це відносно нова мережева архітектура, яка стала найбільш обговорюваною мережевою технологією за останні роки та останньою розробкою в галузі цифрових мереж. Мета даної розробки – зробити ресурси більш керованими, безпечними та контрольованими.

SDN використовується, щоб зробити мережі більш програмованими та полегшити керування мережею. Це дозволяє традиційним мережам розділяти функції рівня керування та даних, створюючи більш динамічний, адаптований, автоматизований та керований дизайн. Мережа програмується за допомогою програмних додатків, які взаємодіють із пристроями рівня даних, які стають основними елементами пересилання та не мають прямого фізичного контролю над ними. Він відокремлює функціональність керування від інструменту мережевої площини та централізує його, маючи ефективне перенаправлення трафіку на відміну від традиційних мереж.

Архітектура програмно-конфігурованої мережі складається з трьох рівнів: додатків, управління та інфраструктури, пов'язаних один з одним через відкриті API-інтерфейси (рис. 1).

Рівень додатків складається зі служб і програм, які мережа надає користувачеві, обробляє статистичні дані про стан мережевих елементів, забезпечує

автоматизовану зміну протоколів при налаштуванні мережі, віртуалізовує мережеві функції та інше.

Рівень керування містить контролер, який є основою цієї технології, оскільки він відповідає за маршрутизацію, управління та контрольні рішення для всіх функцій мережі.

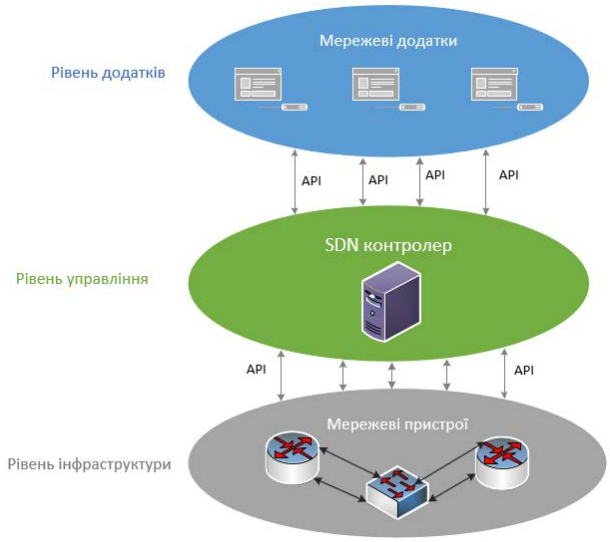

Рис. 1. Загальна архітектура SDN

Рівень керування містить контролер, який є основою цієї технології, оскільки він відповідає за маршрутизацію, управління та контрольні рішення для всіх функцій мережі. Цей рівень взаємодіє з прикладним рівнем через інтерфейс прикладного програмування (API). Рівень інфраструктури складається з мережевих пристроїв (маршрутизаторів або комутаторів), які отримують команди від рівня керування та виконують їх.

Найбільш складні завдання в мережах SDN стоять перед рівнем управління. Вирішення цих завдань покладається на контролер, основним елементом якого є мережева операційна система.

Розглянемо особливості побудови контролера Ryu, протоколи та інтерфейси, функціональний склад елементів. Також побудуємо мережу SDN з використанням контролера Ryu та емулятора mininet, в цій топології потрібно перевірити працездатність контролера та провести тестування параметрів функціонування мережі, для цього проведемо тестування пропускної здатності каналів зв'язку у напрямку від h11 до h9. Для цього будемо передавати UDP трафік при різній пропускній здатності каналів зв'язку і подивимося на поведінку контролера при зміні пропускної здатності.

Аналіз останніх досліджень і публікацій. На сьогоднішній день ведеться досить багато досліджень та розробок що до побудови та використання даної технології в проектах. Різні автори пропонують дуже багато рішень що до використання даної технології в яких одна з найбільших проблем це визначити який контролер краще використати в тих чи інших умовах.

Опубліковано ряд документів, що описують принципи побудови і функціонуваннямереж SDN.

У роботі [1–4] розглянуто загальні вимоги, системні підходи для архітектури мереж SDN. Наведено приклади структурної схеми загального виду контролерам SDN, також розглянуто різновиди контролерів та описано компоненти та структурні схеми контролерів та порівняльна характеристака ідеалізованою структурною схемою мережі SDN.

[5] У цьому документі представлено як порівняння на основі функцій, так і аналіз продуктивності найбільш часто використовуваних реалізацій контролерів Ryu та POX. Порівнюється їх пропускна здатність і затримка в мережевих топологіях на основі простого дерева, повного дерева та традиційної IP-мережі. Представлено що продуктивність контролера Ryu/POX залежить від багатьох різних факторів: апаратного забезпечення контролера та конфігурації алгоритму керування, базової мережевої інфраструктури, кількості комутаторів OpenFlow, кількості хостів, кількості потоків.

[6] У цьому документі аналізується оцінка продуктивності через контролер RYU, враховуючи один комутатор OpenFlow і три вузли. Для трьох різних шляхів між вузлами ми отримали результат після обширного дослідження моделювання, яке оцінювалося за параметрами пропускної здатності,часу проходження, тремтіння та втрати пакетів.

У роботах [7–11] описано основні компоненти та характеристики контролера Ryu. Також відображено архітектуру контролера та загальні принципи його роботи. Написано програму Ryu, яка змусить комутатори OpenFlow працювати як комутатори рівня 2. Відображено перспективи розвитку даного контролера та його майбутню популярність.

[12] У цьому дослідженні пропонується архітектура для ефективної передачі даних датчиків у мережах IoT на основі SDN. У запропонованій моделі було розраховано середню втрату пакетів і затримку, а також досліджено зміни відповідно до кількості перемикань і кількості переходів. Для ефективної комунікації було помічено, що кількість переходів була найефективнішою змінною, але кількість переходів не мала лінійної залежності від загальної кількості комутаторів у мережі. Розташування датчиків виявилося важливим для ефективного зв'язку. Рекомендовано використо-

вувати мережеві моделі, розробляти послуги IoT найкращим чином, контролювати мережу для втручання в різні ситуації, розробляти мережу за допомогою повномасштабного моделювання та робити необхідні виправлення.

У роботі [13] наведено загальні характеристики по вибраним критеріям таким як архітектура, модульність, мова програмування, маштабованість, інтерфейси, стійкість та інші. На основі даних критеріїв розглянуто найпопулярніші контролери (Open Network Operation System (ONOS), OpenDayLight (ODL), OpenKilda, Ryu and Faucet) та порівняння їх між собою оцінкою та відображенням результатів в таблиці.

Метою статті є розкриття особливостей побудови контролера Ryu, опис позитивних та негативних сторін в порівнянні з іншими контролерами та спроба побудувати мережу в емуляторі mininet з використанням даного контролера та зняти характеристики пропускної здатності при різних типах навантаження.

Виклад основного матеріалу. Як і у випадку з більшістю платформ, існують компроміси, які слід враховувати при порівнянні централізованої, тісно пов'язаної площини керування з децентралізованим, масштабованим і слабо пов'язаним альтернативним контролером SDN.

Централізовані архітектури, такі як ONOS і ODL, як правило, легші в обслуговуванні та забезпечують меншу затримку між тісно пов'язаними південними API і північними API. Однак із збільшенням масштабу централізовані контролери можуть стати вузьким місцем. У контексті SD-WAN це може збільшити затримку рівня керування, але його можна пом'якшити в розподіленій архітектурі.

Розподілені архітектури, як в контролерах OpenKilda та Faucet, як правило, складніші для підтримки та розгортання, але можуть дозволити платформі ефективніше масштабуватись. Завдяки відокремленню обробки трафіку, телеметрії та південного інтерфейсу кожну функцію можна масштабувати незалежно, щоб уникнути вузьких місць продуктивності. Крім того, стають життєздатними спеціалізовані інструменти для роботи з великими наборами даних, базами даних часових рядів або масштабним обчисленням шляху без негативного впливу на продуктивність південного протоколу.

Ryu відрізняється від інших варіантів, хоча, маючи базовий набір програм, які запускаються як «платформа», його краще розглядати як набор інструментів, за допомогою якого можна створити функціональність контролера SDN.

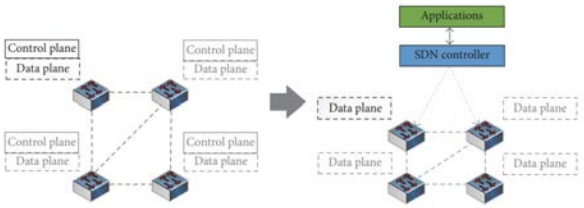

Рис. 2 Приклад розподіленого та централізованого управління мережею SDN

Модульність і розширюваність

Модульність кожного контролера залежить від дизайну та мов програмування. Такі платформи, як ONOS і ODL, мають вбудовані механізми для підключення кодових модулів за рахунок централізації обробки для кожного контролера. Ці два контролери на основі Java використовують переваги контейнерів OSGi для завантаження пакетів під час виконання, що дозволяє дуже гнучкий підхід до додавання функціональності.

Контролери на основі Python, такі як Ryu, надають чітко визначений API для розробників, щоб змінити спосіб керування та налаштування компонентів.

Додавання функціональності до Faucet і OpenKilda досягається шляхом модифікації систем, які використовують їхні північні інтерфейси, наприклад кластер Apache Storm або еквівалент. Це забезпечує додаткову гнучкість використання різних інструментів і мов залежно від проблеми, що вирішується. Крім того, збільшення складності взаємодії на північ не впливає безпосередньо на SDN.

Масштабованість

З розглянутих варіантів тільки ONOS і ODL містять внутрішню функціональність для підтримки кластера. Кожна з цих платформ підтримується розподіленим сховищем даних, яке поділяє поточний стан SDN і дозволяє контролерам перемикатися після відмови у випадку розділу кластера. З появою нових версій кожного з контролерів ця функціональність, схоже, розвивається.

OpenKilda підходить до масштабованості кластера за модульним принципом. У той час як Floodlight використовується як південний інтерфейс для інфраструктури комутатора, відповідальність за PCE і обробку телеметрії переміщується на північ у повністю окремий кластер на основі Apache Storm. Кожен екземпляр Floodlight є ідемпотентним і не вимагає спільного використання стану. Кластер Apache Storm за своєю конструкцією є горизонтально масштабованим і дозволяє збільшити пропускну здатність шляхом додавання вузлів.

Ryu, і Faucet не мають внутрішньої можливості кластеризації та потребують зовнішніх інструментів, таких як Zookeeper, для розподілу бажаного стану. На обох цих платформах додаткові екземпляри контролера можна запускати незалежно, якщо резервна конфігурація залишається ідентичною. Функціональність PCE для цих контролерів може бути переміщена до екземпляра у формі модулів або реалізована подібним чином до OpenKilda, за підтримки обраного кластера обробки.

У міру того, як масштаб SDN зростає, стає неможливим для одного локалізованого кластера обробляти навантаження від кожного комутатора в мережі. Залишаючи осторонь географічний розподіл контролерів, розбиття мережі на менші логічні острівці зменшує потребу в єдиному кластері, орієнтованому на південь, для масового масштабування. Завдяки такому дизайну координація між острівцями стає критичною, і хоча централізований перегляд мережі все ще потрібний, відсутність PCE та обробки телеметрії не повинно впливати на стабільність площини даних після налаштування потоків.

Ryu, Faucet, ODL і ONOS прагнуть масштабувати таким чином, включаючи власні можливості маршрутизації BGP для координації потоків трафіку між островами SDN. Універсальний PCE і обробку телеметрії потрібно буде розробити для кожного з цих випадків з OpenKilda, що забезпечує робочу еталонну архітектуру для досягнення цього. Через стан документації для OpenKilda потрібно буде розробити BGP.

Інтерфейси

З огляду на майбутні вимоги сумісності для керування на південь, ONOS, ODL і Ryu включають протоколи, окрім OpenFlow. P4, Netconf і OF-Config можуть увімкнути додаткові апаратні параметри комутатора, якщо це буде потрібно.

Північний API виявляється однією з ключових відмінностей між пропонованими платформами. ONOS і ODL пропонують найбільший набір північних інтерфейсів із gRPC і RESTful API (серед інших), що робить їх найпростішими для інтеграції. Ryu та OpenKilda пропонують обмежений RESTful порівняно з ONOS та ODL. Faucet використовує зовсім інший підхід до застосування змін, покладаючись на файли конфігурації для відстеження планованого стану системи замість миттєвих викликів API. Цей підхід вимагатиме зовнішніх інструментів для динамічного застосування конфігурації, але відкриває SDN для адміністрування за допомогою добре зрозумілих конвеєрів CI/CD і тестового обладнання.

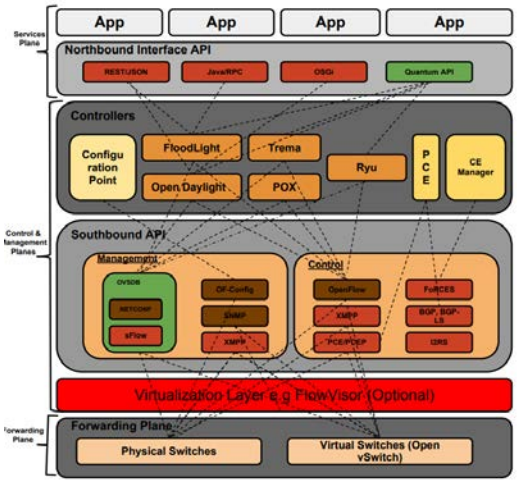

Рис. 3. Інтерфейси контролерів SDN

Телеметрія

Однією з основних проблем із підтримкою SDN є отримання та використання будь-якої доступної телеметрії для визначення стану системи та допомоги у вирішенні проблем. З цього приводу ODL не має функціональних можливостей, оскільки телеметрія все ще є експериментальним модулем в останній версії. ONOS має модулі, які дозволяють використовувати телеметрію через Grafana або InfluxDB.

Faucet може експортувати телеметрію в файли журналу Influxdb, Prometheus або плоский текст. Незважаючи на те, що Prometheus зберігає дані локально, його також можна об'єднати, що забезпечує централізоване агрегування та обробку подій, зберігаючи при цьому локальний кеш для обробки збоїв у обробці вгорі та обслуговування.

OpenKilda використовує Storm, який надає систему обчислень, яку можна використовувати для аналітики в реальному часі. Storm передає дані часових рядів до OpenTSDB для зберігання та аналізу. Neo4j, платформа для аналізу та візуалізації графіків, яка спочатку надавала функції PCE.

Ryu не надає жодної функції телеметрії. Це потрібно забезпечити за допомогою зовнішніх інструментів.

Стійкість

Платформи ONOS і ODL реалізують рідну кластеризацію як частину своїх відповідних пропозицій. ONOS і ODL забезпечують відмовостійкість в системі з непарною кількістю контролерів SDN. У разі виходу з ладу головного вузла новий лідер буде обраний, щоб взяти під контроль мережу. Механізм вибору лідера дещо відрізняється в цих двох контролерах, тоді як ONOS зосереджується на остаточно узгодженому ODL, зосередженому на високій доступності.

Решта контролерів (OpenKilda, Ryu та Faucet) не мають вбудованого механізму кластеризації, натомість покладаються на зовнішні інструменти для підтримки доступності. Це спрощує архітектуру контролерів і звільняє їх від накладних витрат на підтримку розподілених баз даних для інформації про стан. Висока доступність досягається шляхом запуску кількох ідентично налаштованих екземплярів або одного екземпляра, керованого зовнішньою структурою, яка виявляє та перезапускає несправні вузли.

Для Ryu Zookeeper може забезпечити відмовостійкість для моніторингу контролерів, щоб виявити збій контролера та стан шардингу між членами кластера. Зокрема для Faucet, який призначений для розміщення в розподіленій спільній SDN і керування ним за допомогою статичних конфігураційних файлів, перезапуск контролера є швидкою, стабільною вправою, яка не залежить від інфраструктури вищестоящого зв'язку після написання конфігурації.

Мова програмування

ONOS, ODL і OpenKilda написані на Java, для якої на ринку є багато ресурсів для розробки, з хорошою супровідною документацією та доступними бібліотеками. Хоча використання Java не слід розглядати як негатив, процеси Java можуть мати тенденцію бути важкими та вимагати керування ресурсами та конфігурацією, щоб підтримувати їх економічністю та швидко реагувати.

Ryu та Faucet написані мовою Python, яка добре підтримується, і має активну спільноту, що розробляє фреймворк. Документація стисла та технічна, спрямована на розробників, щоб максимально збільшити корисність системи. Python не є швидкою мовою та має невід'ємні обмеження через використання як динамічних представлень типів, так і обмежених багатопоточних можливостей (у порівнянні з Java, Golang або C++).

Controller	Programming language	Northbound APIs
Pox	Python	ad-hoc
Ryu	Python	REST
Nox	C++	ad-hoc
Floodlight	Java	Rest, Java, RPC, and Quantum
Beacon	Java	ad-hoc
OpenDaylight	Java	REST, RESTCONF, XMPP and NETCONF
ONOS	Java	REST and Neutron

Рис. 4. Контролер SDN і мови програмування

Спільнота

І ODL, і ONOS виграють від великих спільнот розробників і користувачів під прапором Linux Foundation Networking. Багато великих міжнародних гравців беруть участь у розробці та управлінні цими проектами, що з часом може збільшити довговічність і безпеку. Можливий недолік полягає в тому, що, як і в будь-якому великому проекті, є багато голосів, які намагаються бути почутими, і на стабільність може вплинути швидкість роботи. Це траплялося з подібними проектами, такими як OpenStack, у найближчому минулому.

OpenKilda – це невелика, але активна спільнота, яка може обмежити підтримку, швидкість і функції платформи. OpenKilda потребує вашої підтримки – спілкуйтеся з нами, щоб взяти участь.

Між цими двома крайнощами знаходяться RYU і Faucet. Обидва цільові контролери добре підтримуються. Завдяки новому характеру галузі, обидва варіанти мають світле майбутнє, з більш простим, раціональним підходом до змін подання та тестування.

Таблиця оцінювання

На основі наведених вище критеріїв ми оцінили кожен продукт за кожним зваженим критерієм. Кожному критерію оцінки було надано максимальну кількість балів відповідно до важливості характеристики і впливу її на розвиток, стабільність і працездатність контролера, наприклад:

- OpenFlow Support – протокол управління процесом обробки даних, що передаються мережею передачі даних маршрутизаторами і комутаторами, що реалізує технологію програмно-конфігурованої мережі. Підтримка протоколів OpenFlow являється одним з важливих факторів на сьогодні.
- Northbound API – є сполучною ланкою між програмами та контролером SDN. Програми можуть повідомляти мережі, що їм потрібно (дані, сховище, пропускна здатність тощо), і мережа може надавати ці ресурси або передавати те, що вона має.
- Southbound API – полегшують контроль над мережею та дозволяють контролеру SDN динамічно вносити зміни відповідно до вимог і потреб у реальному часі.
- Мова програмування – даний критерій впливає на розвиток контролерів, адже це на пряму впливає на швидкість розробки нових можливостей та підтримки стабільності коду самого контролера.
- Та інше…

Відповідно до прикладу наведеного вище було оцінено різні критерії та надано їм відповідні бали для оцінки. Максимальний бал який може набрати контролер буде складати 100 балів і кожній характеристики відповідно до її впливу на контролер надано відповідна кількість балів для

оцінки контролера. В таблиці наведено кількість балів які отримав кожен контролер відповідно до максимуму балів кожної характеристики та виведено загальну кількість балів.

Результати наведено нижче:

Criterion	Weight	ONOS	ODL	OK	Ryu	Faucet
OpenFlow Support	20.0	20.0	19.0	12.0	20.0	20.0
Northbound API support	20.0	20.0	20.0	12.0	16.0	8.0
Southbound API support	10.0	10.0	10.0	6.0	8.0	8.0
Programming Language	5.0	4.0	4.0	4.5	4.5	4.5
Core Components features / services	5.0	4.5	4.5	3.5	2.0	3.5
Native Clustering Capabilities	10.0	9.0	7.0	10.0	2.0	5.0
Typical Architecture	3.0	2.7	2.4	2.7	2.4	2.7
Horizontal Scalability	5.0	3.5	3.0	4.5	1.0	4.0
Vertical Scalability	5.0	3.5	3.0	5.0	4.5	0.5
Extensibility	2.0	1.8	1.6	1.8	1.8	1.5
Community Size & Partnerships	5.0	4.5	4.5	1.0	4.5	3.5
Resilience and Fault Tolerance	5.0	4.0	3.0	4.5	4.0	4.5
Operations Support	5.0	4.5	2.5	4.0	2.5	3.5
Weighted Score	100	92	84.5	71.5	73.2	69.3

Рис. 5. Оцінка контролерів SDN за критеріями

Ці зусилля, витрачені на дослідження поточних платформ програмно-визначених мережевих контролерів (SDN), можуть бути використані, щоб надати користувачам уявлення про доступні контролери SDN з відкритим кодом. Це може допомогти їм вибрати правильний контролер SDN для своєї платформи, який відповідає дизайну мережі та вимогам.

Відповідно до таблиці ми можемо спостерігати позитивні і негативні сторони контролера Ryu відповідно в порівнянні його з іншими контролерами, та перспективні можливості його розвитку.

Якщо ми розглянемо характеристики саме Ryu то відповідно до таблиці ми можемо побачити такі позитивні якості:

- OpenFlow Support – контролер Ryu підтримує протокол OpenFlow. Цей протокол відповідає за управління процесом обробки даних, що передаються мережею передачі даних, маршрутизаторами і комутаторами, що реалізує технологію програмно-конфігурованої мережі. Підтримка протоколів OpenFlow являється одним з важливих факторів на сьогодні.
- Вихідний код Ryu Controller розміщений на GitHub і керується та підтримується відкритою спільнотою Ryu. Завдяки новому характеру галузі, даний контролер має світле майбутнє, з більш простим, раціональним підходом до змін подання та тестування.
- Підтримка Northbound / Southbound API – дані арі полегшують контроль над мережею та дозволяють контролеру SDN динамічно вносити зміни відповідно до вимог і потреб у реальному часі.

Це один із контролерів SDN, спеціально розроблений для забезпечення гнучкості мережі та управління вищою швидкістю трафіку. Ryu включає чітко визначені програмні компоненти разом з API.

Компонентна складова Ryu контролера SDN

Ryu Controller – це відкритий контролер мережі (SDN), призначений для підвищення гнучкості мережі за рахунок посилення управління та адаптації способів обробки трафіку. В загальному, контролер SDN – це мозкова середовище SDN, що передає інформацію про комутатори та маршрутизатори за допомогою південних API, а також про програми та бізнес-логіку за допомогою північних API. Контролер Ryu підтримує NTT і також розвертається в обласних центрах обробки даних NTT.

Контролер Ryu надає програмні компоненти разом із певними інтерфейсами прикладних програм (API), які сприяють розробникам створення нових додатків керування та контролю мережі. Цей компонентний підхід допомагає організаціям налаштувати розвертання відповідно до своїх конкретних потреб; Розробники можуть швидко і легко змінити існуючі компоненти або видалити свої власні, щоб базовий набір відповідав змінним вимогам їх програм.

Вихідний код Ryu Controller розміщений на GitHub і керується та підтримується відкритою спільнотою Ryu. OpenStack, який веде відкриту співпрацю, спрямовану на розробку хмарної операційної системи, яка може контролювати обчислювальні сховища та мережеві ресурси організації, підтримує розгортання Ryu як мережевий контролер.

Весь код Ryu повністю написаний на Python доступний під ліцензією Apache 2.0 і відкритий для всіх. Контролер Ryu підтримує протоколи управління мережею NETCONF і OF-config, а також OpenFlow, який є одним з перших стандартів зв'язку SDN, що найбільш широко використовуються. Він також підтримує розширення Nicira.

IT-адміністратори пишуть спеціальні програми, які повідомляють контролеру Ryu, як управляти комутаторами та маршрутизаторами. Контролер Ryu може використовувати OpenFlow або інші протоколи для взаємодії з площиною пересилання (комутатори та маршрутизатори) для зміни того, як мережа оброблятиме потоки трафіку. Він був протестований та сертифікований для роботи з рядом комутаторів OpenFlow, включаючи Open vSwitch та пропозиції від Centec, Hewlett Packard, IBM та NEC.

Як і будь-який контролер SDN, Ryu також може створювати та надсилати повідомлення OpenFlow, прослуховувати асинхронні події, такі як flow_removed, а також аналізувати й обробляти вхідні пакети. На малюнку 2 нижче зображено компонентну складову фреймворку контролера Ryu:

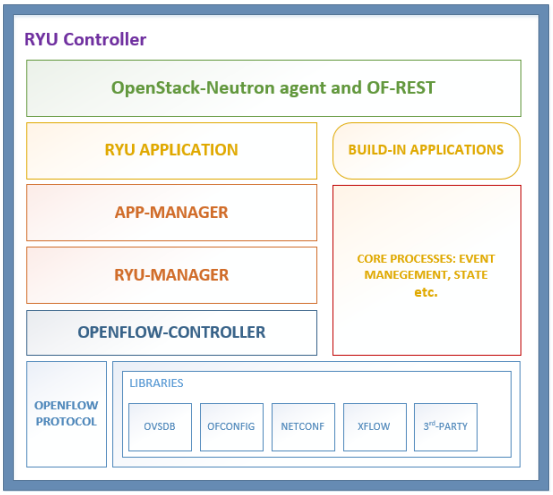

Рис. 6. Функціональна схема контролера RYU

Давайте розглянемо деякі важливі компоненти:

Ryu Libraries. Ryu має вражаючу колекцію бібліотек, починаючи від підтримки відкритих «південних» протоколів SBI і закінчуючи різними операціями обробки мережевих пакетів. Що стосується протоколів SBI, Ryu підтримує OF-Config, Open vSwitch Database Management Protocol (OVSDB), NETCONF, XFlow (Netflow і Sflow) та інші сторонні протоколи.

Протоколи Netflow і Sflow (XFlow) підтримують вибірку та агрегацію пакетів, які в основному використовуються для вимірювання мережевого трафіку.

Протокол OVSDB використовує виклики JSON-RPC для керування фізичним або віртуальним комутатором, який підтримує OVSDB.

У OpenFlow є протокол-розширення OF-CONFIG, що дозволяє керувати основними налаштуваннями комутаторів, такими як черги та стани портів.

NETCONF надає механізми встановлення, керування та видалення конфігурації мережевих пристроїв за допомогою віддаленого виклику процедур RPC. NETCONF використовує XML як засіб надання конфігурації і як формування повідомлень протоколу, який реалізується поверх транспортного.

Third Party protocol – це клієнт-серверний протокол із трьома типами примітивів: Request (використовується клієнтом), Notify (використовується сервером для надсилання інформації про стан клієнту), Responses (надсилається як відповідь на запит).

Бібліотеки сторонніх розробників включають прив'язку Open vSwitch Python, бібліотеку конфігурації Oslo та бібліотеку Python для клієнта NETCONF. Бібліотека пакетів Ryu допомагає аналізувати та створювати різні пакети протоколів, наприклад VLAN, MPLS, GRE тощо.

OpenFlow Protocol. Протокол управління процесом обробки даних, що передаються по мережі передачі даних маршрутизаторами і комутаторами, реалізуючий технологію програмно-конфігурованої мережі. Ryu підтримує протокол OpenFlow до останньої версії 1.4. Він містить бібліотеку кодера та декодера протоколу OpenFlow.

OpenFlow Controller. Крім того, одним із ключових компонентів архітектури Ryu є контролер OpenFlow, який відповідає за керування перемикачами OpenFlow, які використовуються для налаштування потоків, керування подіями тощо. Контролер OpenFlow є одним із внутрішніх джерел подій в архітектурі Ryu.

Managers and Core-Processes. Менеджер Ryu є основним виконуваним файлом. При запуску він прослуховує вказану IP-адресу (наприклад, 0.0.0.0) та вказаний порт (за замовчуванням 6633). Після цього будь-який комутатор OpenFlow (апаратний, Open vSwitch або OVS) може підключитись до диспетчера Ryu. Диспетчер програм є основним компонентом всіх програм Ryu. Усі програми успадковуються від класу RyuApp диспетчера додатків. Компонент ядра процесу в архітектурі включає управління подіями, обмін повідомленнями, управління станом в пам'яті і т. д. Цікаво, що служба обміну повідомленнями Ryu підтримує компоненти, розроблені іншими мовами.

Openstack Neutron agent. На рівні API Ryu включає плагін Openstack Neutron, який підтримує як накладення на основі GRE (протокол тунелювання мережевих пакетів), так і конфігурації VLAN. Ryu також підтримує інтерфейс REST для своїх операцій OpenFlow. Крім того, використовуючи WSGI (фреймворк для з'єднання веб-додатків і веб-серверів на Python), можна легко ввести в програму нові REST API.

APP Manager. Ryu поширюється з кількома додатками, такими як simple_switch, маршрутизатор, ізоляція, брандмауер, тунель GRE, топологія, VLAN і т. д. Програми Ryu є однопотокові об'єкти, що реалізують різні функції. Програми Ryu надсилають одна одній асинхронні події. Кожна програма Ryu має чергу прийому подій, яка в основному являє собою FIFO для збере-

ження порядку подій. Крім того, кожен додаток включає потік для обробки подій з черги. Основний цикл потоку витягує події з черги прийому та викликає відповідний обробник подій. Отже, обробник подій викликається в контексті потоку обробки подій, який працює блокуючим чином, тобто коли обробнику подій передається управління, подальші події для Ryu програми не будуть оброблятися до тих пір, поки управління не буде повернено. Програма Ryu може викликати подію або отримувати подію. Щоб викликати події, програма Ryu викликає відповідні методи ryu.base. app_manager RyuApp, наприклад API send_event або send_event_to_observers.

Архітектура RYU Controller

Контролер RYU забезпечує компоненти програмного забезпечення з чітко визначеними програмними інтерфейсами прикладних програм (API), які полегшують розробникам створення нових додатків для керування та керування мережею.

Контролер RYU SDN має три рівні. Верхній рівень складається з бізнес-програм і програм мережевої логіки, відомих як прикладний рівень. Середній рівень складається з мережевих служб, відомих як рівень керування або структура SDN, а нижній рівень складається з фізичних і віртуальних пристроїв, відомих як рівень інфраструктури. Середній рівень розміщує північні та південні API. Контролер надає відкритий північний API, такий як API керування Restful, REST, API для Quantum, визначений користувачем API через REST або RPC, які використовуються програмами. RYU надає групу компонентів, таких як OpenStack Quantum, Firewall, OFREST тощо, корисних для програм SDN. Метою цих програм є збір мережевих даних за допомогою контролера, виконання аналітики за допомогою алгоритмів, а потім оркестрування нових правил за допомогою контролера. Південний інтерфейс здатний підтримувати кілька протоколів, таких як OpenFlow, Netconf, OF-config тощо. RYU використовує OpenFlow для взаємодії з площиною пересилання (комутаторами та маршрутизаторами), щоб змінити спосіб обробки потоків трафіку в мережі. Він був протестований і сертифікований для роботи з кількома комутаторами OpenFlow, включаючи OpenvSwitch і пропозиції від Centec, Hewlett Packard, IBM і NEC.

Структура Ryu відрізняється від інших рішень тим, що вона надає просту допоміжну інфраструктуру, яку користувачі платформи повинні написати для використання на власний розсуд. Хоча для цього потрібний досвід розробки, це також забезпечує повну гнучкість рішення SDN. Існуючі компоненти можна швидко і легко оновлювати і об'єднувати в існуючі мережі, щоб задовольнити потреби різних програм, що змінюються, використовуючи ці компоненти. Завдяки своїм характеристикам Ryu є відмінним рішенням для невеликих комерційних та експериментальних застосувань. Цей рівень керування дозволяє створювати програми та модулі, оскільки він розроблений на Python.

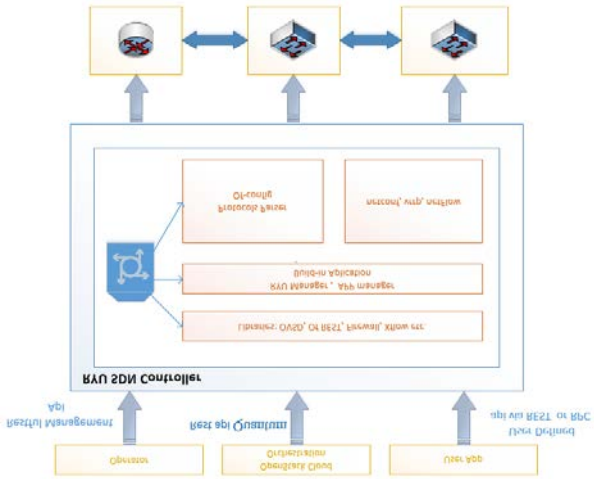

Рис. 7. Архітектура RYU Controller

Оскільки Mininet являється загальною платформою-емулятором, то при запуску мережі без зазначення мережевих параметрів вона не відповідає дійсній SDN мережі. Тому з початку потрібно зазначити низку параметрів і провести декілька кроків налаштування, щоб мережа відповідала вимогам реальної SDN мережі. В нашому випадку буде використовуватись топологія мережі (рис. 1), яка написана на вищому рівні Mininet API (Python API). Топологія Mininet описана в окремому файлі розширення ".py" з залученням сторонніх бібліотек для більш гнучких налаштувань мережі. В роботі була використана бібліотека TCLink, яка надає змогу задавати пропускну здібність, затримку, джиттер у каналах зв'язку, а також RemoteController для описання ONOS контролера, якій в подальшому буде підключений до мережі в якості централізованого органу управління.

Спробуємо побудувати топологію SDN мережі в емуляторі mininet з використанням контролера Ryu та провести тестування параметрів функціонування мережі, для цього проведемо тестування пропускної здатності каналів зв'язку у напрямку від h11 до h9. Для цього, організована передача UDP трафіку при різній пропускній здатності каналів зв'язку.

Дана топологія буде виглядати таким чином:

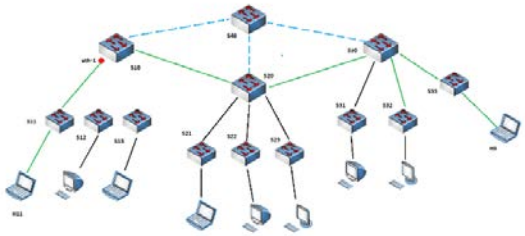

Рис. 8. Топологія SDN мережі

Для такої топології потрібно написати конфігураційний файл мережі з вказанням кількості хостів, свічів та запустити та підєднати до цієї топології контролер Ryu.

Конфігураційний файл мережі/топології має наступний вигляд:

Рис. 9. Приклад конфігураційного файлу

Зазначений код описує процес запуску основних елементів мережі. Цей процес складається з наступних етапів:

– імпорт бібліотек python, що надають змогу компілятору python розуміти синтаксис опису саме топології Mininet;

– додання мережевих елементів: комутаторів з підтримкою OpenFlow, хостів, а також задання їм унікальні адреси та ідентифікатори в мережі, описання розташування контролеру та порт на якому він приймає включення;

– додання мережевих зв'язків – підключення комутаторів між собою, а також підключення хостів до комутаторів;

– запуск мережевих компонентів, Mininet консолі, а також підключення кожного комутатора до контролера Ryu.

Потрібно скачати з GitHub файли контролера та встановити його на ОС, інструкція по встановленні описана в репозеторії який вигружаємо з сайту.

Для того щоб запустити контролер потрібно виконати команду в консолі:

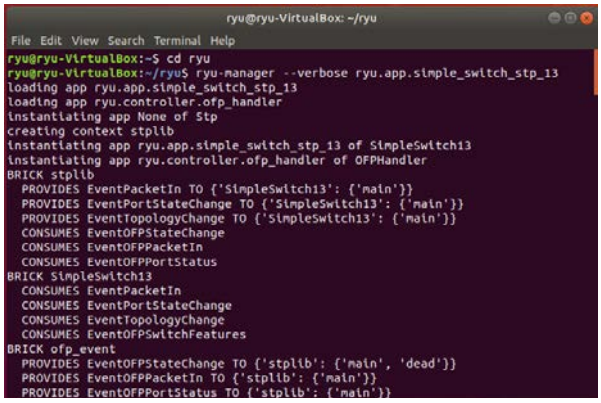

Рис. 10. Приклад запуску контролера Ryu

Після запуску контролера потрібно запустити саму мережу для цього виконаємо команду для запуску нашого файлу:

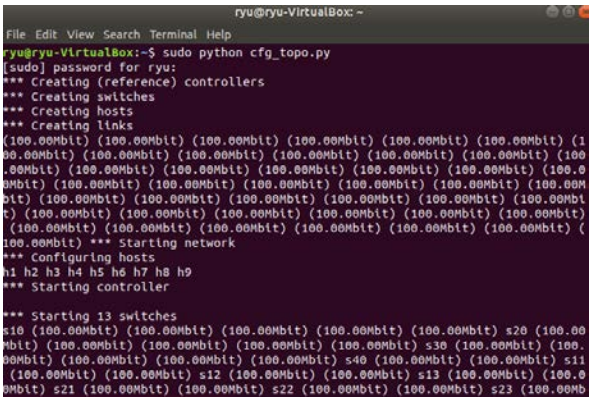

Рис. 11. Приклад запуску мережі в mininet

Після запуску мережі необхідно провести тестування параметрів функціонування мережі. В роботі проведено тестування пропускної здатності каналів зв'язку у напрямку від h11 до h9. Для цього, організована передача UDP трафіку при різній пропускній здатності каналів зв'язку. При цьому були використані наступні вихідні данні:

– топологія мережі, що зображена на рис. 1;

– інформаційний напрямок зв'язку між хостами h11 – h9;

– кількість проміжних вузлів(комутаторів) – 3;

– навантаження в пакетах, що передається – 500 Мбайт;

– швидкість передачі у лініях зв'язку – 100 Мбіт/с.

Для тестування використаємо Iperf та Xterm можливості, де відкриємо консолі на хосты h11 та H9 і за допомогою xterm та запустимо команди для тестування, а саме:

- Сервер: iperf –s –u –p 15 –i 1
- iperf -c 10.0.0.9 –p 15 –n 500M –b 10M –u

У ході проведення досліду було знято часові діаграми швидкості передачі UPD трафіку. Крок підвищення пропускної здантності у лініях зв'язку становить: 10%, 50%, 100% від максимально заданої швидкості передачі лінії зв'язку. Тобто 10, 50, 100 Мбіт/с, відповідно. Часові діаграми наведені на рис. 2.

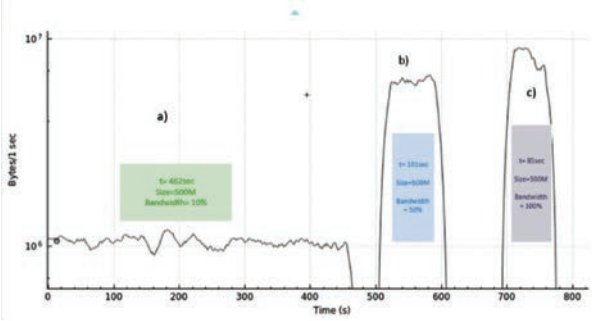

Рис. 12. Швидкість передачі трафіку каналом зв'язку при: а) пропускна здатність 10%; б) пропускна здатність 50%; в) пропускна здатність 100%

Проведені дослідження дозволяють визначити де які особливості. Побудова мережі SDN довільної топології вимагає додаткового написання програмного коду на мові python, а також використання дефольного контролера ryu, який написаний на мові python і можна скачати з їх github репозеторія. Це дає можливість створювати необхідні зв'язки як між елементами мережі, так і між елементами керування мережею.

Висновки. Ryu – це відкритий контролер мережі (SDN), призначений для підвищення гнучкості мережі за рахунок посилення управління та адаптації способів обробки трафіку. Контролер Ryu надає програмні компоненти разом із певними інтерфейсами прикладних програм (API), які сприяють розробникам створення нових додатків керування та контролю мережі. Цей компонентний підхід допомагає організаціям налаштувати розвертання відповідно до своїх конкретних потреб; Розробники можуть швидко і легко змінити існуючі компоненти або видалити свої власні, щоб базовий набір відповідав змінним вимогам їх програм. У майбутньому планується вивчити більше контролерів, щоб зменшити їх вплив на продуктивність SDN.

В даній роботі було наведено порівняльну характеристуку даного контролера з іншими та визначено його позитивні якості в порівнянні з іншими.

- Підтримка протоколу OpenFlow.
- Підтримка Northbound / Southbound API.
- Вихідний код Ryu Controller розміщений на GitHub і керується та підтримується відкритою спільнотою Ryu.
- Ryu контролер написаний на мові програмування Python яка підтрмується і використовується великою спільнотою що дає змогу залучити розробників в спільноту для покращення функціональності і внесення нових ідей в реалізацію даного контролера.

В роботі, використовуючи побудовану топологію та перераховані вихідні дані, було знято часові діаграми при навантаженні напрямку зв'язку. Відображені результати показали, що при пропускній спроможності лінії зв'язку до 50%, передача відбувається з постійною швидкістю без суттєвих змін, що не можна сказати при пропускній спроможності більше 50%. В такому випадку, характер швидкості непередбачуваний, хоча в обох випадках, переданий об'єм трафіку доставляється без втрат. Треба відмітити суттєвий недолік проведених досліджень на базі цих програмних продуктів.

Список літератури:

1. N. McKeown, "OpenFlow: Enabling Innovation in Campus Networks". in SDN: Software Defined Networks, O'Reilly Media, 2013, pp. 85-108. URL: https://www.oreilly.com/library/view/sdn-software-defined/9781449342425/ch04.html

2. Arshad, M. A. Khan, T. A. Zia, and U. Farooq, "Performance Evaluation of SDN Controllers in Wireless Network," URL: https://www.researchgate.net/publication/338071223_Pe rformance_Evaluation_of_SDN_Controllers_i_Wireless_Network

3. F. Lahmar and F. Lahmar, "Controllers in SDN: A Review Report," URL: https://www.researchgate.net/publication/325706050_Controllers_in_SDN_A_Review_Report.

4. "SDN controller (software-defined networking controller)," TechTarget. URL: https://www.techtarget.com/searchnetworking/definition/SDN-controller-software-defined-networking controller.

5. M. Ali and H. Shamim, "Performance Analysis of Ryu-POX Controller in Different Tree-Based SDN Topologies," URL: https://www.researchgate.net/publication/354230822_Performance_Analysis_of_Ryu_POX_Controller_n_Different_Tree-Based_SDN_Topologies.

6. M. Uddin, M. Ali, and M. A. Rahman, "Node to Node Performance Evaluation through RYU SDN Controller," URL: https://www.researchgate.net/publication/338703576_Node_to_Node_Performance_Evaluation_througRYU_SDN_Controller/figures?lo=1&utm_source=google&utm_medium=organic

7. J. Hurwitz, "SDN Series Part IV: Ryu, a Rich Featured Open Source SDN Controller Supported by NTT Labs," URL: https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-opensource-sdn_controller-supported-by-ntt-labs/

8. "Comparison of Software Defined Networking (SDN) Controllers – Part 5: Ryu," Aptira. URL: https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-5-ryu/.

9. "Topics related to Ryu SDN Controller," GitHub. URL: https://github.com/topics/ryusdn_controller

10. "Ryu SDN Framework Documentation," URL: https://ryu.readthedocs.io/en/latest/.

11. D. O'Riain, "RYU SDN Controller: Soft Testbed," URL: https://www.obriain.com/training/sdn/RYU_Soft_Testbed_v2.0.pdf

12. M. Ali, H. Shamim, and M. A. Rahman, "Implementation of an SDN-Based IoT Network Model for Efficient Transmission of Sensor Data," URL: https://www.researchgate.net/publication/337439497_Implementation_of_an_SDN_Based_IoT_Networ_Model_for_Efficient_Transmission_of_Sensor_Data

13. "Comparison of Software Defined Networking (SDN) Controllers – Part 7: Comparison and Product Rating," Aptira. URL: https://aptira.com/comparison-of-software-defined-networking-sdncontrollers-part-7-comparison-and-product-rat

Romanov O.I., Burlaka H.Yu. SDN NETWORK MANAGEMENT USING RYU CONTROLLER

To date, it has been found that the most difficult tasks in SDN networks are faced by the management level. Solving these tasks depends on the controller, the main element of which is the network operating system. This article discusses the construction features of the Ryu controller, protocols and interfaces, and the functional composition of elements. The comparative characteristics of this controller with others are presented and evaluated according to certain criteria, thanks to which you can get an overall picture of the positive and negative sides of this controller. This article reveals the principles of building a mininet network using this controller.

This article describes what the Ryu controller is, namely an open network controller (SDN), designed to increase network flexibility by strengthening management and adapting traffic processing methods. In general, an SDN controller is the brain of SDN, communicating information about switches and routers through southbound APIs, and applications and business logic through northbound APIs. The Ryu framework differs from other solutions in that it provides a simple supporting infrastructure , which users of the platform must write for use at their own discretion. While this requires development expertise, it also provides the full flexibility of an SDN solution. Existing components can be quickly and easily upgraded and integrated into existing networks to meet the changing needs of different applications using these components.

The article examines the components of the Ryu controller, its positive and negative sides in comparison with other controllers. Features of Ryu controller construction, protocols and interfaces, functional composition of elements are considered. An SDN network was also built using the Ryu controller and the mininet emulator, in this topology the performance of the controller was tested and the parameters of the network functioning were tested, for this the bandwidth of the communication channels in the direction from h11 to h9 was tested.

Key words: *mininet, Ryu Controller, OpenFlow, ONOS, APP Manager, Ryu Libraries, bandwidth.*