УДК 621.39
DOI https://doi.org/10.32782/2663-5941/2024.1.1/09

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»

ВЗАЄМОДІЯ КОНТРОЛЕРА RYU З КОМУТАТОРАМИ SDN

У світі технологій мережевого управління контролер Ryu вибивається серед інших, відкриваючи шлях інноваційним застосуванням у сфері програмно-визначених мереж (SDN). Його гнучкість та розширювані можливості роблять його не лише потужним інструментом для створення та керування мережами, але і платформою для реалізації передових концепцій у сфері комунікацій.

Ця стаття досліджує взаємодію між контролером SDN (Software-Defined Networking) Ryu та комутаторами у відкритому середовищі мережі. У статті розглянуто основні аспекти керування правилами OpenFlow, використовуючи контролер Ryu, і розкрито важливі етапи з'єднань між контролером та комутаторами.. У статті розглянуто особливості побудови контролера Ryu, протоколи та інтерфейси, функціональний склад елементів та наведено приклади як можна отримати певну інормацію по різним елементам мережі з використанням OpenFlow.

Стаття присвячена 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.

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.

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

Постановка проблеми. У світі сучасних мережевих технологій поняття SDN (Software-Defined Networking, Програмно-визначена мережа) вже давно не є новизною. Цей підхід до мережевого управління революціонізував спосіб, яким мережі налаштовуються та керуються. SDN розриває традиційні обмеження мережевих інфраструктур, надаючи можливість централізованого керування мережею та відокремлення управління від фізичних пристроїв. Однак основну роль у реалізації SDN відіграє SDN-контролер, який відповідає за збір, обробку та розподіл інструкцій у мережі. У цьому контексті контролер Ryu стає центральною фігурою у світі SDN, завдяки своїм функціональним можливостям та відкритості для розширень. Він відкриває перед собою безмежні можливості для створення різноманітних мережевих рішень та оптимізації мережевого трафіку.

В даній статті ми зосередимось на одному з найпопулярніших та потужних SDN-контролерів, а саме – контролері Ryu.

Контролер Ryu, який розробляється як вільне програмне забезпечення, став неодмінною складовою в світі програмно-визначених мереж. Цей контролер забезпечує взаємодію з комутаторами SDN, реалізуючи важливі функції керування мережею, встановлення правил маршрутизації та обробки трафіку.

У цій статті ми розглянемо ключові аспекти взаємодії контролера Ryu з комутаторами SDN. Ми розкриємо основні принципи комунікації між цими компонентами, дослідимо протокол OpenFlow як мову обміну даними, та проаналізуємо роль Ryu у реалізації програмно-визначених мереж.

Таким чином, ми розпочнемо наше подорож у світі віртуалізації мережі, де контролер Ryu відіграє ключову роль у досягненні гнучкості, масштабованості та ефективності мережевого управління.

Мета даної роботи – розібратися в архітектурі контролера Ryu та принципах взаємодії даного контролера з OpenFlow, розглянути можливості нала-

штування та отримання інформації по контролеру в мережі SDN з використанням емулятору Mininet.

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 використовувати 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 та спроба побудувати мережу в емуляторі mininet з використанням даного контролера.

Виклад основного матеріалу. Протокол OpenFlow є критичним стандартом у світі програмно-визначених мереж (SDN), і контролер Ryu взаємодіє з комутаторами саме через цей протокол. В даному розділі ми розглянемо ключові аспекти протоколу OpenFlow та його ролі у взаємодії з контролером Ryu.

Сутність OpenFlow:

OpenFlow – це відкритий протокол, який розробляється і підтримується Open Networking Foundation (ONF). Він служить мостом між централізованим контролером (яким є контролер Ryu) і комутаторами, які можуть бути розроблені різними виробниками та виконувати відповідні реалізації протоколу.

Архітектура OpenFlow:

Протокол OpenFlow має просту архітектуру, що складається з контролера, який керує комутаторами. Контролер відправляє команди (повідомлення) на комутатори, які виконують ці команди відповідно до правил маршрутизації, встановлених контролером. Ця централізована архітектура дозволяє здійснювати гнучке управління мережею.

Роль контролера Ryu:

Контролер Ryu діє в якості розуміючого агента мережі. Він відправляє запити на налаштування комутаторів, встановлює правила маршрутизації, а також аналізує стан мережі. Протокол OpenFlow дозволяє контролеру Ryu динамічно керувати комутаторами, вносячи зміни в мережеву топологію та маршрутизацію.

Взаємодія за допомогою повідомлень:

Взаємодія між контролером Ryu і комутаторами відбувається через обмін повідомленнями OpenFlow. Контролер відправляє запити, наприклад, запити на додавання правил маршрутизації або видалення їх, і комутатори виконують ці запити та відправляють відповіді назад контролеру.

Підтримка версій:

OpenFlow має кілька версій, і контролер Ryu підтримує різні версії протоколу. Це дозволяє використовувати різні функціональні можливості та оптимізовані реалізації в залежності від конкретних потреб мережі.

Завдяки протоколу OpenFlow, контролер Ryu може забезпечувати централізоване та гнучке управління мережею, що робить його ідеальним інструментом для програмно-визначених мереж і дозволяє мережевим адміністраторам легко налаштовувати та оптимізовувати свої мережі.

Архітектура та компоненти контролера Ryu

Контролер Ryu має гнучку та розширювану архітектуру, яка дозволяє розробникам створювати різноманітні додатки та модулі для програмно-визначених мереж (SDN). Давайте подивимося на ключові складові цієї архітектури:

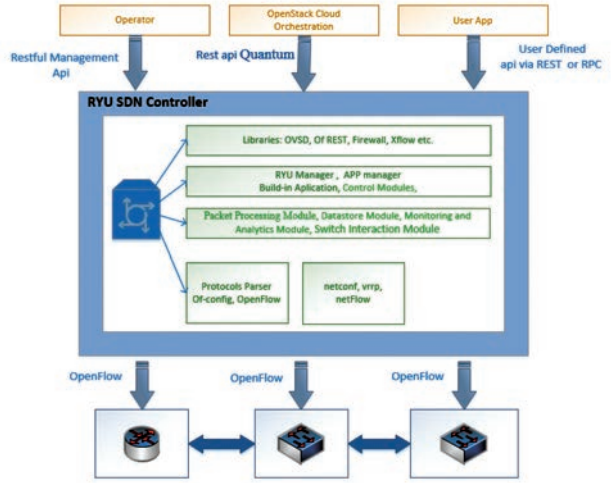

Рис. 1. Архітектура контролера Ryu

Керовані модулі (Control Modules). Це фундаментальні складові архітектури контролера Ryu, які відповідають за обробку різних аспектів управління мережею в середовищі програмно-визначених мереж (SDN). Кожен керований модуль виконує конкретну функцію та реалізує певну логіку обробки даних та подій. Керовані модулі реалізовані у вигляді окремих програмних компонентів або модулів, що дозволяє їх додавати та розширювати незалежно один від одного. Ця модульна структура робить контролер Ryu дуже гнучким та розширюваним і дозволяє розробникам додавати нові функції та розширювати можливості системи без значних змін в існуючому коді.

Керовані модулі можуть виконувати різні функції, включаючи обробку маршрутизації, управління мережевими подіями, аналіз мережевого трафіку, обробку протоколів та багато іншого. Наприклад, модуль маршрутизації може визначати оптимальний шлях для передачі пакетів, в той час як модуль аналізу трафіку може виявляти аномалії в мережі.

Розробники можуть створювати власні керовані модулі та додавати їх до контролера Ryu за допомогою відповідного API. Це дозволяє розширювати функціональність контролера та адаптувати його до конкретних потреб мережі. Керовані модулі можуть взаємодіяти між собою для вирішення складних мережевих завдань. Наприклад, модуль маршрутизації може використовувати дані від модуля аналізу трафіку для прийняття рішень про оптимальний маршрут.

Керовані модулі діють під керівництвом централізованого контролера Ryu, який координує їх роботу та приймає рішення щодо управління мережею.

Приклади керованих модулів в контролері Ryu включають модуль маршрутизації, модуль аналізу трафіку, модуль обробки подій та багато інших. Кожен з цих модулів відповідає за певний аспект управління мережею та допомагає забезпечити гнучкість та функціональність контролера Ryu у середовищі SDN.

API для розширення. Це програмний інтерфейс, який надається контролером Ryu для того, щоб розробники могли створювати власні додатки (або плагіни) для розширення функціональності контролера. Основні аспекти API для розширення:

- Гнучкість та розширюваність: API для розширення робить контролер Ryu дуже гнучким та розширюваним. Розробники можуть створювати власні додатки, які взаємодіють з контролером через це API, додаючи нові можливості та функції до системи.
- Стандартизація інтерфейсу: API для розширення має чіткий та стандартизований інтерфейс, який описує, як взаємодіяти з контролером та як використовувати його функціональність. Це дозволяє розробникам легко розуміти, як користуватися API та як взаємодіяти з контролером.
- Підтримка різних мов програмування: API для розширення контролера Ryu підтримує різні мови програмування, зокрема Python, яка є осно-

вною мовою для розробки додатків для контролера Ryu. Це дозволяє розробникам використовувати мову програмування, з якою вони найкраще знайомі.

• Доступ до функціональності контролера: API для розширення надає доступ до різних функцій та служб контролера Ryu. Це включає в себе можливість додавати правила маршрутизації, обробляти мережеві події, аналізувати мережевий трафік, змінювати конфігурацію мережі та інше.

• Розширення можливостей SDN: API для розширення допомагає розширювати можливості SDN в цілому. Розробники можуть створювати додатки, які вирішують конкретні завдання та вимоги мережі, додавати нові функції та забезпечувати інновації в області програмно-визначених мереж.

• Забезпечення безпеки та автентифікації: API для розширення також може включати механізми безпеки та автентифікації, щоб забезпечити безпечну взаємодію між додатками та контролером.

Підтримка різних протоколів. У контексті програмно-визначених мереж (SDN) і контролера Ryu, підтримка різних протоколів означає, що цей контролер може взаємодіяти з різними комутаторами та мережевим обладнанням, що підтримує різні мережеві протоколи і стандарти. Основні аспекти підтримки різних протоколів:

• OpenFlow: OpenFlow є ключовим мережевим протоколом у світі SDN. Контролер Ryu підтримує версії OpenFlow, такі як OpenFlow 1.0, 1.3 і інші, що дозволяє йому взаємодіяти з комутаторами, які підтримують ці версії протоколу.

• NETCONF: NETCONF є іншим протоколом, який використовується для конфігурації та управління мережевим обладнанням. Контролер Ryu може підтримувати NETCONF для інтеграції зі сховищами конфігурації мережі.

• BGP (Border Gateway Protocol): Контролер Ryu може бути налаштований для роботи з протоколом маршрутизації BGP, що дозволяє керувати маршрутами в мережі та забезпечувати зв'язок з іншими мережевими областями.

• REST API: Контролер Ryu може використовувати RESTful API для взаємодії з іншими додатками та системами через HTTP-запити. Це дозволяє створювати інтегровані рішення та додатки, які взаємодіють з контролером через стандартні HTTP-запити та відповіді.

• SNMP (Simple Network Management Protocol): Для моніторингу та управління мережевим обладнанням, яке підтримує SNMP, контролер Ryu може використовувати цей протокол для отримання статистики та інформації про мережу.

• Інші протоколи: Крім вищезазначених, контролер Ryu може бути налаштований для роботи з іншими мережевими протоколами та стандартами в залежності від конкретних потреб мережі.

Підтримка різних мережевих протоколів дозволяє контролеру Ryu бути універсальним інструментом для управління різними типами комутаторів і обладнанням у різних мережевих середовищах. Вона дозволяє побудовувати багатофункціональні та гнучкі мережі, а також інтегрувати SDN-рішення в існуючі мережеві інфраструктури.

Модуль обробки пакетів (Packet Processing Module). Модуль обробки пакетів є однією з ключових складових архітектури контролера Ryu в системі програмно-визначених мереж (SDN). Він відповідає за обробку та аналіз мережевих пакетів, які прокладають свій шлях через комутатори в мережі.

Основні аспекти модуля обробки пакетів:

• Перехоплення та аналіз пакетів: Модуль обробки пакетів може перехоплювати мережеві пакети, що проходять через комутатори, та аналізувати їх заголовки та вміст. Це дозволяє контролеру Ryu приймати рішення щодо маршрутизації, фільтрації, пересилання пакетів та інших мережевих операцій.

• Прийняття рішень на основі правил: Модуль обробки пакетів використовує набір правил, які визначають, як слід обробляти певні типи пакетів. Ці правила можуть бути встановлені адміністратором мережі або генеруватися автоматично на основі політики мережі.

• Маршрутизація та пересилання: Модуль обробки пакетів може визначати оптимальний шлях для пересилання пакетів в мережі. Він враховує інформацію про маршрутизацію та стан мережі для прийняття рішень про маршрут пакета.

• Фільтрація та політика безпеки: Модуль може використовувати правила фільтрації для відокремлення пакетів, які не відповідають політиці мережі, а також для застосування політики безпеки, включаючи виявлення та блокування потенційно небезпечних пакетів.

• Взаємодія з іншими модулями: Модуль обробки пакетів може взаємодіяти з іншими керованими модулями контролера для вирішення складних мережевих завдань. Наприклад, він може використовувати модуль маршрутизації для визначення оптимального маршруту для пакета.

• Оптимізація мережевого трафіку: Модуль може використовувати аналіз мережевого трафіку для виявлення паттернів та можливостей оптимізації трафіку, таких як згортання пакетів або видалення надлишкових повідомлень.

• Діагностика та моніторинг: Модуль може збирати статистику та інформацію про мережевий трафік для цілей моніторингу та діагностики мережі.

Модуль обробки пакетів грає критичну роль у функціонуванні контролера Ryu та управлінні мережею в середовищі SDN. Він дозволяє контролеру приймати розумні рішення щодо маршрутизації, безпеки та оптимізації мережевого трафіку на основі поточних потреб мережі і політики управління.

Модуль зберігання даних (Datastore Module). Модуль зберігання даних є важливою складовою архітектури контролера Ryu в системі програмно-визначених мереж (SDN). Його основна функція полягає в збереженні та управлінні інформацією, яка стосується мережевого стану, конфігурації та статистики. Основні аспекти модуля зберігання даних:

• Зберігання мережевого стану: Модуль зберігає інформацію про стан мережі, включаючи топологію, статус комутаторів, маршрути, VLAN, адреси MAC і багато іншої інформації.

• Конфігурація мережі: Він також зберігає дані про конфігурацію мережі, такі як правила маршрутизації, правила фільтрації, параметри якості обслуговування (QoS), VLAN-и, IP-адреси і інші параметри мережі.

• Статистика мережі: Модуль може збирати статистику про мережевий трафік, яка може бути використана для моніторингу та аналізу використання мережі, виявлення проблем та оптимізації ресурсів.

• Історія подій: Деякі модулі зберігання даних також можуть зберігати історію подій, яка відстежує, що відбувалося в мережі в минулому. Це може бути корисно для відновлення подій та відлагодження проблем.

• Інтерфейс для отримання та зміни даних: Модуль зазвичай надає програмний інтерфейс для контролера та інших модулів для отримання та зміни даних в збереженому стані мережі.

• Синхронізація з комутаторами: Модуль може взаємодіяти з комутаторами для синхронізації мережевого стану та конфігурації, забезпечуючи єдність управління мережею.

• Захист та безпека даних: Модуль повинен забезпечувати захист конфіденційної інформації та запобігати несанкціонованому доступу до даних.

• Модуль зберігання даних грає важливу роль у впорядкуванні та керуванні мережевими ресурсами в середовищі SDN. Він допомагає контролеру Ryu забезпечити надійне та ефективне функціонування мережі, забезпечуючи доступ до актуальних даних про стан мережі та конфігурацію.

Модуль моніторингу та аналізу (Monitoring and Analytics Module).

Контролер Ryu надає можливості моніторингу та аналізу мережевого трафіку. Цей модуль дозволяє відстежувати стан мережі у реальному часі, виявляти аномалії та здійснювати мережевий аналіз.

Основні аспекти модуля моніторингу та аналізу:

• Збір даних: Модуль активно збирає дані про мережевий трафік, включаючи інформацію про пакети, їхні заголовки, джерела і призначення, порти, протоколи та інші характеристики.

• Аналіз та обробка даних: Зібрані дані проходять через аналізатор, який виконує обробку даних та виявляє важливі паттерни, аномалії та потенційні проблеми.

• Моніторинг мережевого стану: Модуль відслідковує стан комутаторів, підключених до мережі, стан з'єднань, пропускну здатність та інші параметри для нагляду за працездатністю мережі.

• Генерація звітів та відомостей: Модуль може створювати звіти та відомості про використання мережі, стан комутаторів, аналіз трафіку та інші аспекти мережевої діяльності.

• Візуалізація даних: Зібрані та оброблені дані можуть бути візуалізовані у вигляді графіків, діаграм, теплових карт і інших графічних елементів для полегшення розуміння та аналізу мережевого стану.

• Виявлення аномалій і проблем: Модуль може автоматично виявляти аномалії та проблеми в мережі, такі як витоки трафіку, перевищення обсягу мережевого навантаження і т. д.

• Інтеграція з іншими модулями: Зібрані дані можуть бути використані іншими модулями контролера для прийняття рішень щодо управління мережею.

Модуль моніторингу та аналізу є критично важливим для вирішення завдань моніторингу, аналізу та діагностики в сучасних програмно-визначених мережах. Він допомагає адміністраторам мережі відстежувати мережевий стан, виявляти проблеми та оптимізувати ресурси для забезпечення надійності та ефективності мережі.

Модуль взаємодії з комутаторами (Switch Interaction Module). Основна функція цього модуля – взаємодія з комутаторами через протокол OpenFlow або інші мережеві протоколи. Він відповідає за встановлення правил маршрутизації, обробку подій від комутаторів та надсилання команд комутаторам.

Основні аспекти модуля взаємодії з комутаторами:

• Підключення до комутаторів: Модуль ініціює підключення до фізичних або віртуальних комутаторів в мережі, встановлюючи зв'язок із ними через протоколи зв'язку, такі як OpenFlow.

• Посилання на комутатори: Модуль зберігає інформацію про кожен підключений комутатор, включаючи ідентифікатори, IP-адреси, порти, стан з'єднань та інші параметри.

• Керування правилами пересилання: Модуль взаємодіє з комутаторами для встановлення правил пересилання пакетів та потоків даних. Він передає інструкції комутаторам щодо того, як обробляти мережевий трафік відповідно до політики мережі.

• Оновлення конфігурації: Модуль може оновлювати конфігурацію комутаторів, включаючи VLAN-и, параметри якості обслуговування (QoS), правила безпеки та інші налаштування.

• Моніторинг та відлагодження: Модуль дозволяє контролеру відстежувати стан комутаторів, визначати події, які стосуються комутаторів, і відлагоджувати можливі проблеми.

• Синхронізація стану: Модуль може синхронізувати стан мережі та конфігурацію між контролером і комутаторами, щоб забезпечити єдність управління мережею.

• Обробка подій від комутаторів: Модуль взаємодіє з подіями, які відправляються комутаторами, і реагує на них відповідно до політики мережі та потреб.

• Оптимізація пересилання: Модуль може оптимізувати пересилання мережевого трафіку, встановлюючи правила пересилання, які дозволяють ефективно використовувати ресурси мережі.

Модуль взаємодії з комутаторами є ключовим для забезпечення взаємодії контролера Ryu з фізичними та віртуальними комутаторами в мережі. Він дозволяє контролеру керувати мережею та впливати на шляхи пересилання пакетів для досягнення бажаних мережевих цілей.

Узагальнюючи, архітектура контролера Ryu дозволяє створювати гнучкі та розширювані програмно-визначені мережі, а також сприяє активному розвитку та інноваціям у світі SDN.

Взаємодія Контролера Ryu з комутаторами за допомогою OpenFlow. Спробуємо побудувати топологію SDN мережі в емуляторі mininet з використанням контролера Ryu. Для такої топології потрібно написати конфігураційний файл мережі з вказанням кількості хостів, свічів та запустити та підєднати до цієї топології контролер Ryu.

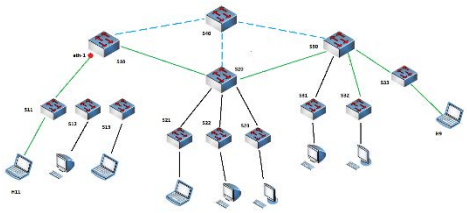

Рис. 2. Топологія SDN мережі

Конфігураційний файл мережі/топології має наступний вигляд:

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.link import TCLink
from mininet.node import Controller, OVSSwitch, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel

def multiControllerNet():
    net = Mininet( controller=Controller, switch=OVSSwitch, link=TCLink )
    print '*** Creating (reference) controllers'
    c1 = RemoteController( 'c1', ip='127.0.0.1', port=6633 )
    #c2 = RemoteController( 'c2', ip='172.17.0.6', port=6633 )
    #c3 = RemoteController( 'c3', ip='172.17.0.7', port=6633 )
    print '*** Creating switches'
    s10 = net.addSwitch( 's10', dpid='0000000000000010', mac='00:00:00:00:SW:10' )
    s20 = net.addSwitch( 's20', dpid='0000000000000020', mac='00:00:00:00:SW:20' )
    ...
    print '*** Creating hosts'
    h1 = net.addHost( 'h1', ip='10.0.0.1/24', mac='00:00:00:00:0H:01' )
    h2 = net.addHost( 'h2', ip='10.0.0.2/24', mac='00:00:00:00:0H:02' )
    ...
    print '*** Creating links'
    #net.addLink( s10, s20, bw=100 )
    net.addLink( s10, s40, bw=100 )
    net.addLink( s20, s30, bw=100 )
    net.addLink( s30, s40, bw=100 )
    net.addLink( s10, s11, bw=100 )
    ...
    print '*** Starting network'
    net.start()
    c1.start()

    s10.start( [ c1 ] )
    s20.start( [ c1 ] )
    s30.start( [ c1 ] )
    ...
    CLI( net )
if __name__ == '__main__':
    setLogLevel( 'info' )  # for CLI output
    multiControllerNet()
```

Рис. 3. 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:

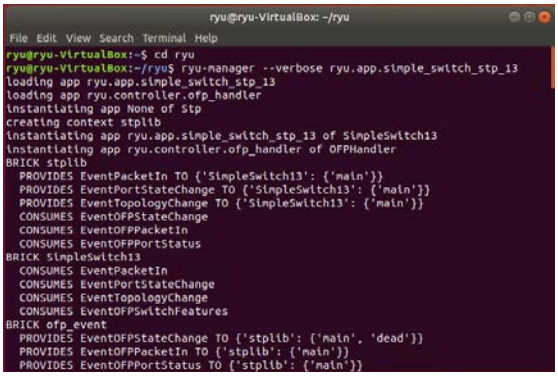

Рис. 4. Приклад запуску контролера Ryu.

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

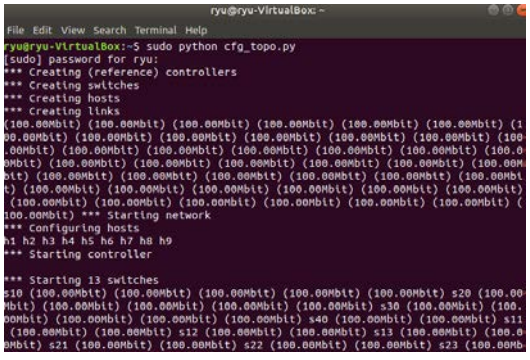

Рис. 5. Приклад запуску мережі mininet

В окремій консолі можна отримати різну інформацію по комутаторам.

sudo ovs-ofctl --protocols OpenFlow13 dump-desc s10 – Зі списку комутаторів можна отримати загальний опис кожного за допомогою «утиліти керування комутаторами OpenFlow» [14].

```
ryu@ryu-VirtualBox:~$ sudo ovs-ofctl --protocols OpenFlow13 dump-desc s10
OFPST_DESC reply (OF1.3) (xid=0x2):
Manufacturer: Nicira, Inc.
Hardware: Open vSwitch
Software: 2.9.8
Serial Num: None
DP Description: s10
```

Рис. 6. Утиліти керування комутаторами OpenFlow

Якщо потрібно дізнатися детальний опис конкретного комутатора потрібно виконати команду **sudo ovs-ofctl --protocols OpenFlow13 show s10:**

```
ryu@ryu-VirtualBox:~$ sudo ovs-ofctl --protocols OpenFlow13 show s10
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:0000000000000010
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 1(s10-eth1): addr:4e:33:8d:a6:cd:98
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s10-eth2): addr:d2:6c:d0:e8:8c:71
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s10-eth3): addr:0e:f6:b7:06:f5:ff
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 4(s10-eth4): addr:86:85:01:70:57:89
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 5(s10-eth5): addr:16:03:81:77:ca:4a
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 6(s10-eth6): addr:36:5d:9e:de:b9:f3
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s10): addr:2e:88:f1:ff:e0:47
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x7): frags=normal miss_send_len=0
ryu@ryu-VirtualBox:~$
```

Рис. 7. Детальний опис комутатора s10

Також можна отримати інформацію про порти на конкретному комутаторі **sudo ovs-ofctl --protocols OpenFlow13 dump-ports s10:**

```
ryu@ryu-VirtualBox:~$ sudo ovs-ofctl --protocols OpenFlow13 dump-ports s10
OFPST_PORT reply (OF1.3) (xid=0x2): 7 ports
  port LOCAL: rx pkts=0, bytes=0, drop=198, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=1639.257s
  port  "s10-eth6": rx pkts=35, bytes=3907, drop=0, errs=0, frame=0, over=0, crc
=0
           tx pkts=1041, bytes=68704, drop=0, errs=0, coll=0
           duration=1639.275s
  port  "s10-eth4": rx pkts=40, bytes=4207, drop=0, errs=0, frame=0, over=0, crc
=0
           tx pkts=1039, bytes=68591, drop=0, errs=0, coll=0
           duration=1639.277s
  port  "s10-eth1": rx pkts=145, bytes=13010, drop=0, errs=0, frame=0, over=0, c
rc=0
           tx pkts=936, bytes=59901, drop=0, errs=0, coll=0
           duration=1639.277s
  port  "s10-eth5": rx pkts=40, bytes=4207, drop=0, errs=0, frame=0, over=0, crc
=0
           tx pkts=1037, bytes=68414, drop=0, errs=0, coll=0
           duration=1639.277s
  port  "s10-eth2": rx pkts=45, bytes=4742, drop=0, errs=0, frame=0, over=0, crc
=0
           tx pkts=1033, bytes=67966, drop=0, errs=0, coll=0
           duration=1639.277s
  port  "s10-eth3": rx pkts=857, bytes=53227, drop=0, errs=0, frame=0, over=0, c
rc=0
           tx pkts=222, bytes=19561, drop=0, errs=0, coll=0
           duration=1639.277s
```

Рис. 8. Інформація про порти на комутаторі s10

Дамп потоків OpenFlow для коммутатора **sudo ovs-ofctl --protocols OpenFlow13 dump-flows s10:**

```
ryu@ryu-VirtualBox:~$ sudo ovs-ofctl --protocols OpenFlow13 dump-flows s10
 cookie=0x0, duration=1710.724s, table=0, n_packets=866, n_bytes=51960, priority
=65535,dl_dst=01:80:c2:00:00:00 actions=CONTROLLER:65535
 cookie=0x0, duration=1710.735s, table=0, n_packets=199, n_bytes=16779, priority
=0 actions=CONTROLLER:65535
```

Рис. 9. Дамп потоків OpenFlow для коммутатора s10

Початкове узгодження встановлення зв'язку OpenFlow Як показано на малюнку 6, контролер Ryu та комутатор обмінюються повідомленнями OFPT_HELLO, кожен з яких може визначити версію OpenFlow іншого та працювати з найменшим спільним знаменником. Після цього початкового обміну, кожен пристрій може обмінюватися подальшими повідомленнями з використанням OpenFlow v1.3 [14].

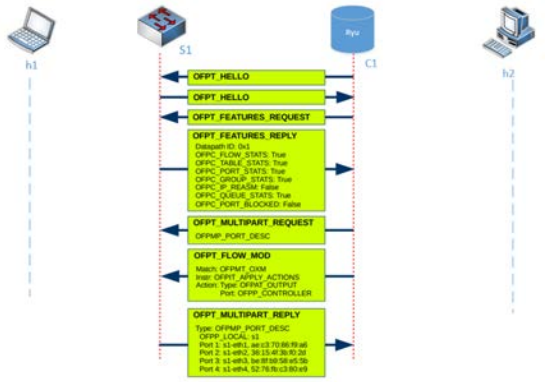

Рис. 10. Початкове рукостискання OpenFlow v1.3 [14]

Потім контролер Ryu відправляє повідомлення OFPT_FEATURES_REQUEST, щоб отримати ідентифікатор каналу даних (DPID) та можливості комутатора. DPID – це 64-бітове число, яке однозначно ідентифікує шлях передачі даних. Молодші 48 біт використовуються MAC-адреси комутатора, а старші 16 біт визначаються розробником, наприклад, для ідентифікатора VLAN. Комутатор відповідає OFPT_FEATURES_REPLY, який включає DPID та можливості, що підтримуються шляхом передачі даних. Потім контролер Ryu відправляє складовий запит на опис портів комутатора, який отримує відповідь з описом всіх портів комутатора, підтримують OpenFlow.

Контролер Ryu також відправляє комутатору модифікацію потоку OpenFlow, яка вказує на будь-який збіг, виведення на порт комутатора, підключений до контролера, іншими словами, відправку на контролер.

Це можна побачити і в OvS. Як тільки OvS отримує модифікацію потоку, він додає її до своєї таблиці потоків.

sudo ovs-ofctl --protocols OpenFlow13 dump-flows s10

cookie=0x0, duration=804.819s,
table=0, n_packets=416, n_bytes=24960,
priority=65535,dl_dst=01:80:c^2:00:00:00
actions=CONTROLLER:65535
cookie=0x0, duration=804.830s, table=0,
n_packets=161, n_bytes=13601, priority=0
actions=CONTROLLER:65535

Рис. 11. Статус порту комутатора

Потім комутатор відправляє повідомлення PFPT_PORT_STATUS контролеру Ryu для кожного зі своїх портів, як показано на малюнку 7. Коли комутатор отримує пакети, наприклад, на малюнку 14, на комутаторі приймається пакет ICMP. Він виконує перевірку запису таблиці і виявляє, що вона має запис «промах таблиці» без потоку. Для обробки умов відсутності таблиці контролер Ryu відправляє запис потоку з знаками підстановки, яка багато в чому аналогічна маршруту за умовчанням в IP-маршрутизації. Вона відповідає всім полям і має пріоритет, рівний 0. Вона називається записом потоку таблиці-промаху і дає команду комутатор OpenFlow пересилати пакети в порт контролера (дія = вихід: порт контролера) [14].

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 статті ми розкрили основні аспекти контролера Ryu , зараз досить тяжко найти корисну інформацію про контролер що відповідає дійсності. Також ми навели приклад як можна запустити даний контролер в побудували топологію і показали яким чином можна перевірити працездатність елементів мережі та описати процес взаємодії контролера з комутаторами за допомогою OpenFlow.

Основна проблема на сьогодні по даному контролеру це обмежена інформація по його роботі, тому в даній статті ми намагалися розкрити базові аспекти для розуміння та початку роботи з цим контролером.

Список літератури:

1. N. McKeown, "OpenFlow: Enabling Innovation in Campus Networks," in SDN: Software Defined Networks, O'Reilly Media, 2013, pp. 85-108. [Online]. Available: 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," [Online]. Available: https://www.researchgate.net/publication/338071223_Performance_Evaluation_of_SDN_Controllers_i_Wireless_Network

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

4. "SDN controller (software-defined networking controller)," TechTarget. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/SDN-controller-software-defined-networkingcontroller.

5. M. Ali and H. Shamim, "Performance Analysis of Ryu-POX Controller in Different Tree-Based SDN Topologies," [Online]. Available: 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," [Online]. Available: https://www.researchgate.net/publication/338703576_Node_to_Node_Performance_Evaluation_throug RYU_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," [Online]. Available: 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. [Online]. Available: https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-5-ryu/.

9. "Topics related to Ryu SDN Controller," GitHub. [Online]. Available: https://github.com/topics/ryusdn_controller

10. "Ryu SDN Framework Documentation," [Online]. Available: https://ryu.readthedocs.io/en/latest/.

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

12. Романов, О., Бурлака, Г., Берестовенко, О., & Підпалий, О. (2023). ТЕХНІЧНІ ОСОБЛИВОСТІ ПОБУДОВИ LI-FI МЕРЕЖІ ЗА ДОПОМОГОЮ МЕТОДІВ КЕРУВАННЯ SDN. Вісник Черкаського державного технологічного університету, (3), 16–25. https://doi.org/10.24025/2306-4412.3.2023.284893

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

14. "Testbed Manual by Diarmuid Ó Briain is licensed under CC BY-SA 4.0," [Online]. Available: https://www.obriain.com/training/sdn/RYU_Soft_Testbed_v2.0.pdf.

15. Romanov, O., Korniienko, N., Obod, I. , Svyd, I. Construction of the SDN Control Level Based on ONOS // UkrMiCo 2021 – 2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics, Proceedings, 2021, страницы 127–132, doi: 10.1109/UkrMiCo52950.2021.9716691.

16. Романов О.І., Нестеренко М.М., Фесьоха Н.О. Аналіз сучасних технологій віртуалізації для побудови інформаційно телекомунікаційних систем //Збірник наукових праць [Військового інституту телекомунікацій та інформатизації].-2019.-Вип. 1.-С. 82-90. Режим доступу: http://nbuv.gov. ua/UJRN/Znpviti_2019_1_13.

17. Romanov,O., Nesterenko, M., Veres,,L. Integration Of Modern Protocols Ip-Telephony In Ims Architecture, 2018 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), Odessa, Ukraine, 2018, pp. 1-4, doi: 10.1109/UkrMiCo43733.2018.9047587.

18. Romanov, O., Korniienko, N., Burlaka, H. Construction of the SDN Transport Network Model using the T-API Interface, 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT), Lviv, Ukraine, 2021, pp. 220-224, doi: 10.1109/AICT52120.2021.9628980.

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

In the world of network management technology, the Ryu controller stands out from the rest, paving the way for innovative applications in the field of software-defined networking (SDN). Its flexibility and extensibility make it not only a powerful tool for creating and managing networks, but also a platform for implementing advanced concepts in the field of communications.

This paper explores the interaction between Ryu's Software-Defined Networking (SDN) controller and switches in an open network environment. The article discusses the main aspects of managing OpenFlow rules using the Ryu controller, and the important stages of connections between the controller and switches are revealed. to various network elements using OpenFlow.

The article is about the Ryu Controller, an open-source network controller (SDN) designed to increase network flexibility by enhancing management and adapting the way traffic is handled. 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 that users of the platform must write to 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. Due to its characteristics, Ryu is an excellent solution for small commercial and experimental applications.

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