Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»

УДК _____

До захисту допущено
В.о. завідувача кафедри

_____ Явіся В.С.___
(підпис) (ініціали, прізвище)

“___”_____2019_р.

Магістерська дисертація

на здобуття освітнього ступеня «магістр»

Спеціальність 172 Телекомунікації та радіотехніка,

(код і назва)

За освітньо-професійною програмою Інженерія та програмування інфокомунікацій.
на тему: «Дослідження методу рішення завдань управління контролером в мережі SDN»
Виконав: студент _2_ курсу, групи ТЗ –з81мп
(шифр групи)

Клець Вадим Петрович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник професор, д.т.н. професор Романов О.І._____ __
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант_____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н, доцент Созонник Г.Д._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 рік

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою Інженерія та програмування інфокомунікацій.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____Явіся В.С.
(підпис) (ініціали, прізвище)
«___»_____2019 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Клець Вадиму Петровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: «Дослідження методу рішення завдань управління контролером в мережі SDN»

науковий керівник дисертації Романов Олександр Іванович д.т.н, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_07_» «_11_» 2019р. № _3854-с_

2. Строк подання студентом дисертації 02.12.2019

3. Об'єкт дослідження: управління елементами мережі SDN

4. Предмет дослідження: метод рішення завдань управління контролером елементами мережі SDN

5. Перелік завдань, які потрібно розробити:
1. структура, елементи мережі SDN, їх призначення та функції;
2. принципи побудови мережі SDN;

3. протоколи мережі SDN;
4. порівняння ефективності «Північних» протоколів;
5. принципи побудови контролера SDN.

6. Орієнтовний перелік ілюстративного матеріалу: діаграма зі значеннями часу, який необхідний для виконання вказаного набору конфігурацій обома протоколами, величини використання пропускної здатності та кількості сигнальних повідомлень для реалізації конкретного сценарію.

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 20.10.2019

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Пошук та аналіз літератури	21.10.2019	
1	Принципи побудови мережі SDN. ЇЇ відмінності від традиційних мереж	20.03.2019	
2	Протоколи мережі SDN	20.06.2019	
3	Принципи побудови контролера. Функціональна схема	20.09.2019	
4	Інтеграція архітектури SDN в існуючу інфраструктуру оператора зв’язку	10.11.2019	
5	Вступ. Висновки	27.11.2019	

Студент _____ Клець В.П.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Романов О.І.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Тема роботи: Дослідження методу рішення завдань управління контролером в мережі SDN.

Текстова частина дипомної роботи: с.107, рис.70, табл.5, джерел 24.

На даний момент існує багато варіантів реалізації концепції SDN для різних сценаріїв використання, тому перед споживачами даного продукту постає складне питання вибору необхідної архітектури SDN, яка задовольнятиме вимоги до конкретної мережі. Для того, щоб обрати підходящу архітектуру необхідно розуміти принципи роботи, побудови та відмінності у роботі різних реалізацій SDN. Ключовим моментом роботи мережі SDN є метод, згідно з яким контролер управляє елементами мережі, адже від нього залежить швидкість реакції елементів на команди контролера, ефективність використання наявних ресурсів мережі, складність інтеграції архітектури SDN з існуючою інфраструктурою і навіть різноманітність доступного функціоналу.

Метою роботи є підвищення ефективності роботи мережі SDN за рахунок реалізації належного методу рішення завдань управління контролером елементами мережі.

Об’єктом дослідження є управління елементами мережі SDN.

Предметом дослідження є метод рішення завдань управління контролером елементами мережі SDN.

В даній роботі було проведено дослідження структури і принципів роботи «Південних» протоколів та ефективності їх роботи шляхом порівняння таких показників, як тривалість виконання операції, ефективність використання наявної ємності каналів, кількість повідомлень, необхідних для виконання певної операції, та робота протоколу при надходженні запиту на зміну конфігурації одного пристрою одночасно від двох додатків.

З результатів дослідження випливає, що протокол NETCONF краще задовольнятиме вимоги мережі, ніж протокол OpenFlow, оскільки потребує менше часу для виконання команд контролера та меншу кількість сигнальних повідомлень.

Результатами даної роботи можуть керуватись оператори чи компанії, які планують інтегрувати архітектуру SDN до своєї мережевої інфраструктури.

SDN, NETCONF, OpenFlow, контролер, API, «Північний» інтерфейс, «Південний» інтерфейс, YANG, SD-WAN, SD-DCN, CloudVPN, IP.

ABSTRACT

Topic: “Investigation of the method to solve the control problems of controller in the Software-Defined Networking”

It contents 107 pages, 70 figures, 5 tables and 24 sources.

There are many options for implementing the SDN concept for different usage scenarios, so the consumers of this product face the difficult choose of the SDN architecture that will meet the requirements of a specified network. In order to choose the right architecture, it is necessary to understand the working principles, construction and differences in the operation of different SDN implementations. The key to the operation of the SDN network is the method by which the controller manages the network elements, as it depends on the response time of the elements to the controller commands, the efficiency of using available network resources, the complexity of integrating the SDN architecture to the existing infrastructure and even the variety of available functionality.

The purpose of this thesis is to increase the efficiency of the SDN network through the implementation of the proper method of solving the control problems of controller.

The object of this investigation is a managing of the elements in the SDN network.

The subject of the investigation is the method to solve the control problems of controller in the SDN network.

This thesis investigated the structure and principles of the Southbound protocols and their performance by comparing such indicators as the duration of the operation, the efficiency of the available bandwidth capacity, the number of messages required to perform a certain operation, and the operation of the protocol when the configuration change requests for one device are simultaneously sent by two applications.

The results of the investigation show that NETCONF will better meet network requirements than OpenFlow because it takes less time to execute controller commands and need fewer alarm messages.

The results of this thesis can be used by operators or companies that want to integrate the SDN architecture into their network infrastructure.

SDN, NETCONF, OpenFlow, controller, API, Northbound interface, Southbound interface, YANG, SD-WAN, SD-DCN, CloudVPN, IP.

ЗМІСТ

РЕФЕРАТ 4

ABSTRACT 6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 11

ВСТУП 13

1. ПРИНЦИПИ ПОБУДОВИ МЕРЕЖІ SDN. ЇЇ ВІДМІННОСТІ ВІД ТРАДИЦІЙНИХ МЕРЕЖ 21

1.1. Принципи побудови традиційних мереж 21

1.2. Структура, елементи мережі SDN, їх призначення та функції 24

1.3. Принципи побудови мережі SDN 27

Висновки до розділу 1 31

2. ПРОТОКОЛИ МЕРЕЖІ SDN 33

2.1. Протокол OpenFlow 33

2.1.1. Базова архітектура протоколу OpenFlow 33

2.1.2. Таблиці протоколу OpenFlow 38

2.1.3. Типи повідомлень протоколу OpenFlow 46

2.2. Протокол NETCONF 47

2.2.1. Рівні протоколу NETCONF 51

2.2.2.Можливості протоколу NETCONF

					КПІ ім.Ігоря Сікорського _3854_-с 04.ТЗ-з81мп.2019.ПЗ		
Змн.	Лист	№ докум.	Підпис	Дата			
Розроб.		Клець В.П.			Дослідження методу рішення завдань управління контролером в мережі SDN	Літ.	
Перевір.		Романов О.І.				Арк. 8	Аркушів 122
Реценз.		Созонник Г.Д.				ІТС	
Н. Контр.		Петрова В.М.					
Затверд.		Явіся В.С.					

2.3.Порівняння ефективності роботи протоколів NETCONF та OpenFlow 61

2.3.1.Потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача Б 71

2.3.2.Потік високого пріоритету з ємністю каналу 20Гбіт/с для користувача А 72

2.3.3. Потік високого пріоритету для обох користувачів зі взаємним поділом 82

2.3.4. Запит на смугу шириною 40 Гбіт/с 85

2.3.5.Терміновий запит на смугу шириною 100 Гбіт/с 92

2.3.6. Стрес-тест 95

Висновки до розділу 2 97

3. ПРИНЦИПИ ПОБУДОВИ КОНТРОЛЕРА.ФУНКЦІОНАЛЬНА СХЕМА 101

3.1. Контролер OpenDayLight 102

3.2. Контролер OpenFlow 107

3.3. Контролер Huawei 111

3.4. Контролер NetCracker 112

Висновки до розділу 3 113

4. ІНТЕГРАЦІЯ АРХІТЕКТУРИ SDN В ІСНУЮЧУ ІНФРАСТРУКТУРУ ОПЕРАТОРА ЗВ’ЯЗКУ 114

4.1. Канал точка-точка 114

4.2. Доступ до мережі Інтернет 115

4.3. Інтеграція до RAN-мережі оператора 116

Висновки до розділу 4 117

ЗАГАЛЬНІ ВИСНОВКИ 118

ПЕРЕЛІК ПОСИЛАНЬ 120

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ISP	Internet Service Prodiver
SDN	Software-Defined Networking
BGP	Border Gateway Protocol
vIMS	Virtual IP Multimedia Subsystem
VoLTE	Voice over LTE
vCPE	Virtual Customer Premises Equipment
MIB	Management Information Base
OSI	Open System Interconnection
CLI	Command Line Interface
RFC	Request for Comments
IETF	Internet Engineering Task Force
SPF	Shortest Path First
MPLS TE	Multiprotocol Label Switching Traffic Engineering
IGP	Interior Gateway Protocol
EGP	Exterior Gateway Protocol
OSPF	Open Shortest Path First
ISIS	Intermediate System to Intermediate System
API	Application Programming Interface
FIB	Forwarding Information Base
XML	eXtensible Markup Language
TCP	Transmission Control Protocol
SSH	Secure Shell

RPC	Remote Procedure Call
BoD	Bandwidth on Demand
CMF	Configuration Management Framework
ONF	Open Network Foundation
TTP	Table Type Patterns
ASIC	Application-Specific Integrated Circuit
AAA	Authentication Authorization and Accounting
REST	REpresentational State Transfer
WWW	World Wide Web
HTTP	HyperText Transfer Protocol
MD-SAL	Model Driven – Service Abstraction Layer
DOM	Document Object Model
LAN	Local Area Network
DHCP	Dynamic Host Configuration Protocol
IP	Internet Protocol
DNS	Domain Name System
VRF	Virtual Routing and Forwarding
OSS	Operations support system
RAN	Radio Access Network
E2E	Edge-to-Edge
NMS	Network Monitoring System

ВСТУП

Конвергенція інформаційних та комунікаційних технологій є основим вектором розвитку операторів більшості країн світу. Віртуалізація та хмарні обчислення стали основою їх стратегії створення глобально пов'язаного світу, в якому всі комунікаційні функції будуть відокремлені від спеціалізованого апаратного обладнання та переміщені до «хмарного середовища», де їх можна централізовано виконувати та контролювати.

Очікується, що основна мережа буде відповідно змінена та адаптована. Нова мережа буде чутливою до потреб додатків та адаптуватиметься відповідно до їхніх вимог без викриття її внутрішніх характеристик (наприклад, протоколів, топології, схеми адресації). SDN - це наступний крок в еволюції мережевих технологій до підтримки конвергенції інфокомунікаційних технологій.

Сучасні IP-мережі складаються з безлічі різних вузлів (маршрутизаторів, комутаторів), кожен з яких має власну незалежну площину управління, яка в свою чергу управляє локальною площиною передачі даних (у тому ж фізичному шасі) та забезпечує обмін маршрутною інформацією для реалізації маршрутизації пакетів. Ця парадигма є ключовою в концепції сучасних IP-мереж: модель розподіленої площини управління забезпечує високу надійність і гарантує постійну доступність і працездатність мережі. Враховуючи цю властивість IP-мережі до автоматичної маршрутизації, легко зрозуміти, чому розгортання SDN наразі повільніше, ніж очікувалося. SDN замінює цю базову і головну перевагу IP-мереж на повністю централізоване керуванням мережею і викликає певні побоювання щодо надійності SDN.

Переваги SDN не можна оцінити лише з перспективи маршрутизації. SDN дає можливість програмувати мережу та впроваджувати новий рівень автоматизації в процес управління мережею. Якщо врахувати велику кількість ручних процесів, які притаманні при розгротанні та експлуатації типової IP-мережі (налаштування/обслуговування/управління мережею), SDN вносить

зміни, аналогічні з впровадженням механічного ткацького верстата в англійській промисловій революції кінця 18-го століття, коли ручні операції були замінені на механічні. Це означає, що SDN підвищить ефективність роботи мережі, знизивши «виробничі» витрати.

Основна концепція SDN, закладена командою доктора Скота Шенкера полягала в тому, щоб спростити управління мережі, її конфігурацію та експлуатацію, які зараз реалізуються на складних машинних пропрієтарних мовах. З впровадженням елементу програмування в мережу, її можна розглядати як абстрактну, замість групи незалежно налаштованих вузлів.

На Рисунку 1 зображено дві моделі мережі з першої презентації SDN доктором Шенкером.[1]

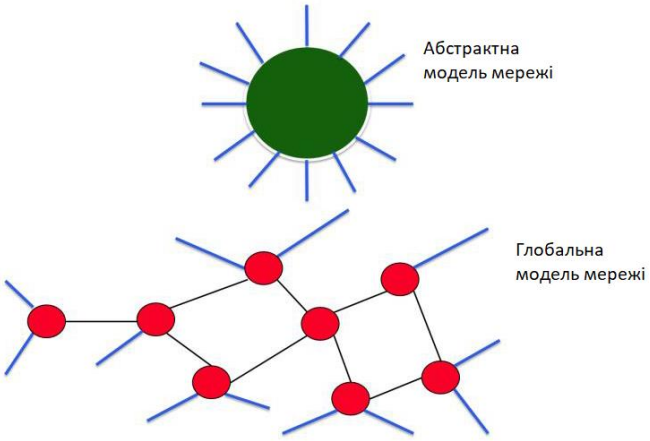

Рисунок 1. Абстрактна та глобальна моделі мережі.

Згідно з новою абстрактною моделлю, мережею можна керувати як одним і простим цілим, а не як складною сукупністю вузлів і протоколів, що притаманно глобальній моделі. Мережі стають подібними до сервісу онлайн-карт: користувач вибирає пункти відправлення та прибуття (А і Б), конкретну послугу (пішохідний рух, рух на автомобілі, рух на громадському транспорті) та конкретний показник (найкоротший маршрут, найшвидший маршрут). Мережа SDN створює безпосередній зв'язок між А і Б, притримуючись вказаних показників якості та вибирає шлях, виходячи з фактичного стан руху (перехрестя, світлофори, умови руху автомагістралей, тощо) без потреби налаштування кожного окремого сегмента шляху.

Оскільки SDN є новою технологією, то уявлення відносно неї часто є хибними. Найбільш поширені міфи описані нижче.

Хибний міф №1: Для розгортання SDN потрібно модернізувати всю мережу. Деякі компанії не наважуються переходити на архітектуру SDN, оскільки вважають, що для цього їм потрібно буде повністю замінити існуюче обладнання. В даному випадку необхідно аналізувати мережу конкретного оператора, оскільки SDN вимагає підтримку «Південних» протоколів мережевими елементами, таких як NETCONF або OpenFlow. Сценарії, в яких частина мережевих елементів не підтримує даних протоколів, також мають стратегію переходу до архітектури SDN, наприклад шляхом встановлення нового програмного забезпечення з підтримкою NETCONF або OpenFlow, якщо апаратних ресурсів для цього достатньо.

Хибний міф №2: SDN не готовий до WAN-магістралей. В останні роки мережа Google SDN часто слугувала ідеальним прикладом для демонстрації переваг конепції SDN. Google, звичайно, є першопрохідцями, оскільки вони використовують SDN для реалізації зв’язку між дата-центрами з 2011 року. Якщо ми детальніше ознайомимося з тим, що зробили Google і чому - ми зрозуміємо, що ця модель не зовсім підходить для використання в мережі провайдерів інтернет-послуг. Google потребували надшвидку мережу, здатну робити масштабні обчислення маршрутів для трафіку типу Захід-Схід (між дата-центрами), що відповідає вимогам живого програмно-розподіленого обчислювального середовища, яке значно відрізняється від ISP. Google реалізували SDN-контроль у власному централізованому механізмі обчислення маршрутів на базі протоколу BGP, оскільки вони зрозуміли, що стандартна платформа для маршрутизації не мала достатньо потужності для обробки та перерахунку маршрутної інформації. Google, звичайно, має найбільш широке розгортання SDN, але в цьому випадку SDN застосовується до мережі, яка не повністю відповідає вимогам оператора магістральної мережі і SDN не може бути рекомендованим, навіть якщо він показує новий рівень контролю трафіку, який не існував досі. Причину, по якій SDN досі не

використовується в магістральних мережах слід шукати в економічному стані, в якому зараз знаходяться оператори. Зацікавлені сторони хочуть отримати швидку окупність всіх ресурсів, вкладених в інфраструктуру, і саме тому основне застосування SDN можна побачити в дата-центрах та на мережах доступу. У цих мережевих секторах оператори можуть швидко створити нові послуги, які можуть бути запропоновані на ринку для швидкого доходу. Кожна інвестиція в автоматизацію магістральних мереж оптимізує внутрішні процеси і, отже, зменшує операційні витрати, але не створює нової послуги і не збільшує доходи за короткий час.

Хибний міф №3: SDN - це фактор блокування виробників. Думка про те, що програмовані мережі будуть позбавлені від закритих мережевих рішень конкретних вендорів частково є правдивими. Насправді за останні роки було реалізовано лише кілька прикладів мульти-вендорних рішень для побудови цілої мережі. Розподіл управління та передачі даних зробить можливим використання обладнання різних виробників в межах однієї мережі, тому що управління буде централізованим. SDN розблоковує розгортання послуг провайдера. Протягом останніх років усі послуги мережі розроблялися на базі загального та однакового функціоналу, розробленого кількома виробниками. SDN мережі відкривають нові можливості на ринку телекомунікацій, де кожен оператор може створити свої нові послуги та відкрито конкурувати з іншими операторами. За допомогою програмованої мережі оператор може краще визначити свою стратегію, звільняючи розвиток власної мережі від залежності від обладнання певного виробника, що прискорює час виходу на ринок з ціллю отримання нових доходів. SDN дає можливість самостійно програмувати мережеву інфраструктуру замість використання функціоналу одного виробника мережевого обладнання.

На Рисунку 2 зображено три основні напрямки використання архітектури SDN.

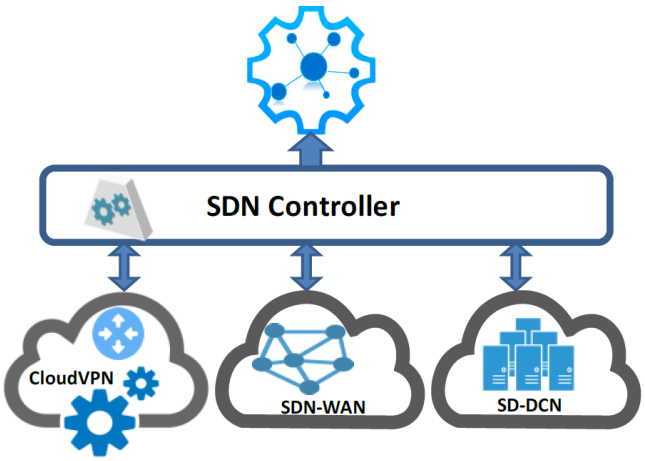

Рисунок 2. Напрямки використання SDN.

Як видно з Рисунку 2, SDN використовується для наступних сценаріїв:

- SD-DCN: SDN використовується для координації роботи оверлейної мережі(віртуальної), тобто сукупності тунелів, що з'єднують усі компоненти хмарної інфраструктури, та для реалізації зв'язку між дата-центрами. У деяких випадках SDN також використовується для управління андерлейною (фізичною) мережею дата-центрів. Інтеграція дата-центрів є однією з головних тенденцій розвитку стратегічних програм на 2020 рік: оператори першого рівня прагнуть сконцентрувати ресурси дата-центрів в меншій кількості місць, тобто, декілька регіональних основних дата-центрів, доповнених кількома місцевими крайовими дата-центрами. Таким чином, SDN розглядається з кількох причин:
    - ввести автоматизацію, яку вимагають сучасні операційні процеси дата-центрів на мережі;
    - підтримка Cloud OS для налаштування мережевої інфраструктури, яку вимагають хмарні програми;
    - координувати зв’язок між датацентрами на основі запитів додатків, наприклад, через такі функціональні можливості, як забезпечення певної пропускної здатності за вимогою або виділення необхідної пропускної здатності у вказані проміжки часу.

Завдяки тиску, який створює необхідність інтергації дата-центрів, використання SD-DCN є частиною першої хвилі застосування технології SDN в телекомунікаційних мережах;

- CloudVPN: SDN використовується для розділення фізичних ресурсів мережі на логічні домени. Кожен домен призначається «орендареві», який налаштовує його набір сервісів. SDN також координує відповідні додаткові види обслуговування. Як і для випадків застосування SD-DCN, оператори першого рівня також розглядають рішення віртуальних мереж, орієнтовані на сегмент «Бізнес-бізнес», будуючи VPN-підключення поверх існуючої інфраструктури IP/MPLS. Багатоточкові VPN, з доступом до мережі Інтернет або до дата-центрів є прикладами оверлейних послуг. «Орендарям» дозволяється встановлювати, контролювати та змінювати конфігурацію своїх оверлейних сервісів через портал. Рішення з використання SDN пропонує наступні переваги:
  - можливість «орендаря» самостійно надавати послугу, набагато швидше, ніж за допомогою традиційних L2/L3 VPN;
  - послуги зв'язку можуть бути віртуалізованими та централізованими, що допомагає зменишити операційні витрати;
  - додаткові види послуг (такі як брандмауер, балансир навантаження, послуга виявлення вторгнень) реалізуються як додаткові сервіси та стають доступними для продажу і приносять більший дохід оператору.
- SD-WAN: спрощує управління та експлуатацію WAN. Ця концепція схожа на ту, в якій за допомогою SDN реалізовують технологію віртуалізації для вдосконалення управління та роботи центрів обробки даних. Основною ціллю SD-WAN є дозволити компаніям будувати високоефективні WAN, використовуючи доступ до Інтернету з нижчими витратами, що дозволяє бізнесу частково або повністю замінити більш дорогі приватні технології підключення до WAN, такі як MPLS. Незважаючи на те, що дана технологія є початковою ціллю використання SDN, застосування SD-WAN

потребє більше часу, щоб перейти від стадії лабораторних тестувань до розгортання на комерційній мережі. Замість того, щоб розділити площини управління та передачі даних, оператори першого рівня зосереджуються сьогодні на тому, як застосування SDN дозволить їм досягти більш гнучкого контролю над своїми мережами, забезпечуючи основу для перетворення їх інфраструктури на хмарну. Основна увага приділяється автоматизації налаштування мережевих шляхів після їх розробки мережевими інженерами. SDN, доповнений великим набором функцій управління, таких як мережева аналітика, підтверджує дизайн та перетворює його в набір політик, які передаються мережевим елементам. Суть полягає в тому, щоб покращити спосіб управління трафіком по мережі та оптимізувати використання мережевих ресурсів. Це досягається, наприклад, шляхом реалізації процесу замкнутого циклу, коли контролер SDN надсилає набір політик у мережу, а колектор аналітики даних отримує результати та повертає їх назад до контролера для подальшої оптимізації використання наявних ресурсів. Переваги є досить простими:

- значно покращується рівень автоматизації операційних процесів;
- вводиться програмування на мережевому рівні.

Природне питання, яке виникає в операторів: з чого почати? Відповісти на це питання непросто, адже треба розглядати архітектуру мережі та спектр послуг, що надає оператор, але деякі компроміси можна знайти.

Деякі оператори рівня 2 працюють над тим, щоб перенести функції деяких мережевих елементів на хмарну інфраструктуру, тобто реалізувати віртуалізовані рішення для мобільного ядра (vIMS, VoLTE). В цьому випадку, SDN може координувати розподіл мережевих ресурсів в дата-центрі. Дана архітектура входить до варіанту SD-DCN, про який говорилося вище. Для цього не потрібно великих змін в архітектурі мережі, зокрема, якщо контролер SDN допомагає оператору реалізувати віртуальну комунікаційну інфраструктуру, без впливу на існуючу андерлейну мережу, забезпечуючи

взаємодію з існуючими платформами дата-центрів, а також елементами зі сторони «Північого» інтерфейсу (наприклад, OSS або сервісний оркестратор).

Деякі оператори 2-го рівня надають послуги лише підприємствам. Cloud VPN надає можливості досягти двох головних цілей, про які йшлося раніше: зменшення операційних витрат, пов’язаних з процесом надання сервісу та збільшення загального доходу. Роль SDN залежить від прийнятої конфігурації: для vCPE SDN контролює конфігурацію мережі та управляє як CPE, розгорнутим на сайті підприємства, так і програмною частиною в дата-центрі. Якщо надання послуг також активовано, SDN надає функцію їх налаштування та координації. На цьому етапі важливо виділити те, що фреймворк SDN повинен мати мультивендорну підтримку, оскільки CPE можуть надходити від різних постачальників і також мати відкриті «Північні» інтерфейси для взаємодії з сервісним оркестратором, щоб отримувати інструкції щодо налаштування віртуалізованих функцій.

На даний момент існує багато варіантів реалізації концепції SDN для різних сценаріїв використання, тому перед споживачами даного продукту постає складне питання вибору необхідної архітектури SDN, яка задовольнятиме вимоги до конкретної мережі. Для того, щоб обрати підходящу архітектуру необхідно розуміти принципи роботи, побудови та відмінності у роботі різних реалізацій SDN. Ключовим моментом роботи мережі SDN є метод, згідно з яким контролер управляє елементами мережі, адже від нього залежить швидкість реакції елементів на команди контролера, ефективність використання наявних ресурсів мережі, складність інтеграції архітектури SDN з існуючою інфраструктурою оператора і навіть різноманітність доступного функціоналу.

1.ПРИНЦИПИ ПОБУДОВИ МЕРЕЖІ SDN. ЇЇ ВІДМІННОСТІ ВІД ТРАДИЦІЙНИХ МЕРЕЖ

1.1.Принципи побудови традиційних мереж

Оскільки традиційні мережі будуються по розподіленому принципу, то кожен пристрій у традиційній мережі здійснює і управління мережею, і передачу даних. Тому в традиційних мережах зустрічаються наступні проблеми:

1. складні мережеві протоколи та їх реалізація роблять управління та конфігурацію складними - багато протоколів, стандартів та RFC, багато відмінностей синтаксису командних рядків у різних виробників змушують мережевих інженерів витрачати роки, щоб набути навичок управління та конфігурації мережевих елементів різних виробників;
2. впровадження нових функцій для мережевих елементів відбувається дуже повільно (Рисунок 1.1) - потрібні роки, щоб визначити стандарт для нового рішення в IETF, для нової версії функцій, потрібен час на оновлення обладнання та виконання конфігурації перед розгортанням сервісу;
3. не централізоване управління для обрахування маршрутів трафіку - для традиційних IP-мереж траси обчислюються на основі SPF, визначається наступний хоп, який не може бути змінений, навіть якщо ми знаємо, що на найкоротшому шляху існує перевантаженість мережі, і є альтернативний шлях. MPLS TE також стикається з тією ж проблемою, навіть якщо його можна використовувати для перенаправлення трафіку.

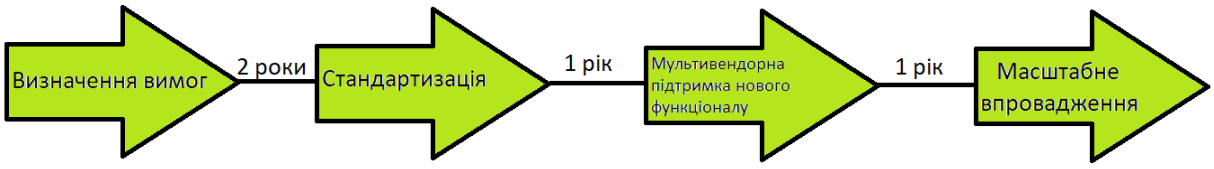

Рисунок 1.1.Процес впровадження нового протоколу.

На Рисунку 1.2 зображено принцип роботи алгоритму SPF для традиційних IP-мереж.

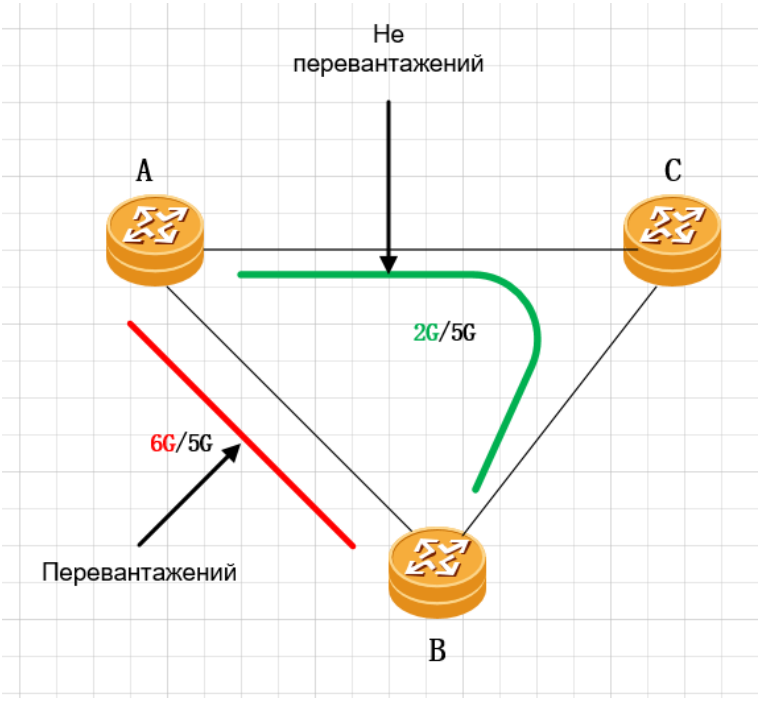

Рисунок 1.2.Принцип роботи SPF алгоритму традиційних IP-мереж.

Згідно з принципом роботи алгоритму SPF, трафік від маршрутизатора A до маршрутизатора B піде по шляху A-B. Наприклад, в певний момент часу необхідна ширина смуги пропускання каналу буде близькою до 6Гбіт/с, а доступна ширина смуги становить 5Гбіт/с і канал стає перевантаженим, в той же час шлях A-C-B завантажений не повністю. Припустимо, що технологію MPLS TE було впроваджено для вирішення проблеми перевантаження каналів, але тунелі MPLS TE конфігуруються заздалегідь і не здатні вирішити проблеми зі сплесками трафіку в режимі реального часу, спричиняючи перевантаженість певних каналів мережі.

Підхід до побудови розподілених мереж спричинив розгортання та налаштування багатьох протоколів для реалізації управління на пристроях, включаючи протоколи IGP, протоколи EGP, протокол MPLS тощо. Організація IETF створила тисячі стандартизованих протоколів, і ця кількість продовжує зростати. Тому мережевий інженер повинен вивчати та засвовати складні технології, щоб розгорати нові мережі, та підтримувати їх працездатність. Також деякі виробники розгортають пропрієтарні протоколи для роботи своїх пристроїв і якщо мережа оператора є мультивендорною, то в

інженерів виникатимуть труднощі в експлуатації та технічному обслуговуванні даних пристроїв.

Якщо розглянути сучасний мережевий пристрій, то він логічно складається з трьох компонентів(Рисунок 1.3):

1. рівень адміністрування. Завдання цього рівня забезпечити керованість пристроєм. Він відповідає за обробку зовнішніх взаємодій користувача та за адміністративні задачі, такі як аутентифікація, ведення логів та конфігурацію через веб-інтерфейс або CLI;
2. рівень управління трафіком - це різні алгоритми і функціонал, завданням якого є автоматична реакція на зміни трафіку, тобто інтелект пристрою. Він відповідає за адміністрування внутрішніх операцій пристроїв, надаючи інструкції, які використовуються для перенаправлення пакетів. Він також запускає протоколи маршрутизації та комутації та подає операційні дані назад до рівня адміністрування;
3. передача трафіку – функціонал, який забезпечує фізичну передачу даних через пристрій, використовуючи дані, що надходять від рівня управління трафіком, для визначення вихідного інтерфейсу.

Рисунок 1.3.Логічна модель традиційних мережевих пристроїв.

Традиційні мережі будуються по розподіленому принципу, де кожен пристрій проводить обчислення самостійно і налаштовується незалежно. Наприклад, трафік входить в елемент А та виходить з елменту В (Рисунок 1.2). Коли А отримає трафік, то перевірятиме відповідно до таблиці маршрутизації.

Базуючись на даних з таблиці маршрутизації, він вирішує, що для того, щоб досягти B, він повинен обрати C в якості наступного хопа. Потім A пересилає трафік до C. Елемент C працює точно так само, як і A: перевіряє таблицю маршрутизації, знаходить інформацію про наступний хоп і передає трафік до B. Такий спосіб перенаправлення трафіку називається перенаправленням на хопі. Таблиця маршрутизації створюється за допомогою статичної маршрутизації або динамічної маршрутизації. У великомасштабних мережах зазвичай використовуються протоколи динамічної маршрутизації, такі як OSPF або ISIS. Кожен маршрутизатор у домені маршрутизації збирає інформацію про стан каналів, а потім незалежно від інших маршрутизаторів, використовуючи той самий алгоритм маршрутизації, обирає найкращий маршрут. Така мережа називається розподіленою мережею.

1.2.Структура, елементи мережі SDN, їх призначення та функції

Архітектуру мережі SDN логічно можна розділити на три рівні (Рисунок 1.4): прикладний рівень, рівень управління та рівень інфраструктури.

Прикладний рівень охоплює рішення, орієнтовані на розширення мережевих служб. Ці рішення в основному є програмними додатками, які взаємодіють з контролером. Відкриті інтерфейси API посилаються на програмні інтерфейси між програмними модулями контролера та додатками SDN. Ці інтерфейси є відкритими для клієнтів, партнерів та спільноти з відкритим вихідним кодом для розвитку та модифікації. Прикладний рівень охоплює велику кількість додатків для задоволення різних потреб клієнтів, таких як автоматизація мережі, гнучкість та програмованість тощо. Деякі домени програм SDN включають інженерію трафіку, віртуалізацію мережі, моніторинг та аналіз мережі, дослідження мережевих служб, контроль доступу тощо. Логіка управління для кожного екземпляра програми може бути запущена як окремий процес безпосередньо на апаратному забезпеченні контролера в кожному домені.

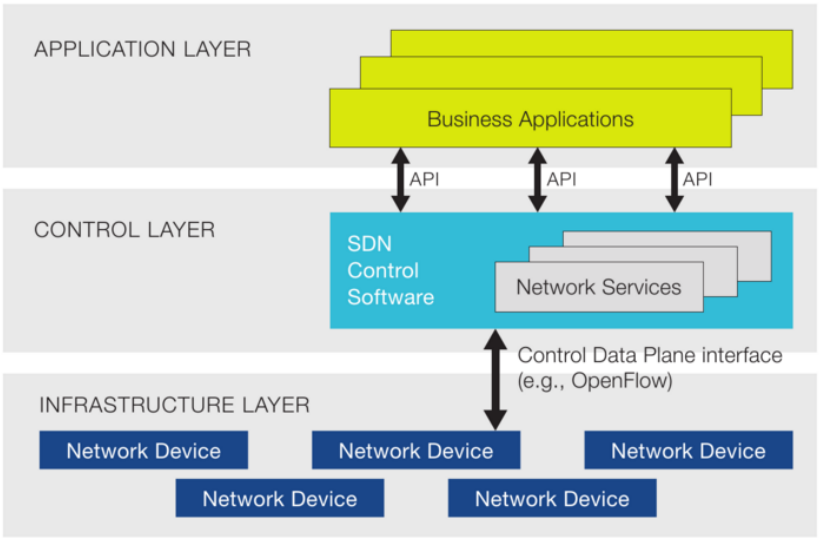

Рисунок 1.4. Загальна архітектура мережі SDN.

Рівень управління включає логічно централізований контролер SDN. Він приймає запити від прикладного рівня через чітко визначені API та здійснює управління та моніторинг мережевих пристроїв за допомогою стандартних протоколів. Він є ядром архітектури SDN і реалізується контролерами кожного домену, які збирають інформацію про фізичний стан ділянки мережі, яка належить кожному контрольному домену.

Рівень інфраструктури складається з фізичного мережевого обладнання, включаючи комутатори та маршрутизатори. Даний рівень забезпечує програмоване і швидкісне обладнання та програмне забезпечення, яке відповідає галузевим стандартам. На нижньому рівні фізична мережа складається з апаратних пристроїв передачі даних, які зберігають таблиці для швидкої передачі пакетів (FIB), а також пов'язані метадані, включаючи пакети, потоки і лічильники портів. Таблиці передачі даних можуть бути базуватись на MAC-адресах(таблиці 2-го рівня) або на IP-адресах(таблці 3-го рівня). Пристрої фізичної мережі можуть бути згруповані в один або більше окремих доменів контролерів, де кожен домен має щонайменше один фізичний контролер. Для реалізації зв’язку між контролером та елементами рівня інфраструктури, такими як фізичні або віртуальні комутатори та маршрутизатори, використовуються «Південні» протоколи. Найбільш

популярними «Південними» протоколами є NETCONF, OpenFlow, PCEP, SNMP, OVSDB.

Архітектура SDN має наступні характеристики:

- прямо програмована, тобто управління мережею безпосередньо програмується, оскільки воно розділене з функціями передачі даних;
- гнучка, тобто абстрагування площини управління від передачі даних дозволяє адміністраторам динамічно регулювати потоки трафіку на всій мережі для задоволення потреб, які змінюються і виникають в певний момент часу;
- має централізоване управління, тобо функції управління зосереджені в програмних контролерах SDN, які підтримують глобальний стан мережі. Для решти елементів мережі SDN, контролер має вигляд звичайного логічного комутатора;
- програмно конфігурована, тобто SDN дозволяє мережевим інженерам налаштовувати, керувати, захищати та оптимізувати мережеві ресурси дуже швидко за допомогою динамічних автоматизованих додатків SDN, які вони можуть писати самі, оскільки додатки не залежать від пропрієтарного програмного забезпечення певного виробника;
- базується на відкритих стандартах та нейтральна до виробиків обладнання. Завдяки тому, що архітектура SDN реалізується на відкритих стандартах, вона значно спрощує мережевий дизайн та взаємодію між елементами, тому що всі інструкції та команди управління надходять від контролера, а не від пристроїв та протоколів певного виробника.

В архітектурі SDN, контролер працює як драйвер для певного пристрою, який налаштовує цей пристрій, читаючи команди від додатків та записуючи їх в пам'ять пристрою. Для андерлейної мережі, контролер - це програмне забезпечення, яке керує та обмінюється всіма ресурсами мережевої інфраструктури з додатками.

Головна ідея концепції SDN полягає в розміщенні мережевої операційної системи між мережевою інфраструктурою та прикладним рівнем. Мережева операційна система є відповідальною за узгодження та управління ресурсами всієї мережі, і зображення в абстрактному уніфікованому вигляді всіх компонентів мережі для додатків, що працюють поверх неї. Концепція є аналогічною принципу роботи типової комп'ютерної системи, в якій операційна система лежить між апаратним забезпеченням та користувачем і відповідає за управління апаратними ресурсами та обслуговування програм користувача. Аналогічно, SDN пропонує логічно централізоване середовище, в якому мережеві адміністратори та розробники можуть програмувати, налаштовувати мережу та керувати нею.

1.3.Принципи побудови мережі SDN

SDN є новим підходом до побудови мереж, в якому фізично розділяють площину управління та площину передачі даних на два різних апаратних засоби. Відмінності реалізації управління елементами мережі між традиційними IP-мережами та SDN зображено на Рисунку 1.5.

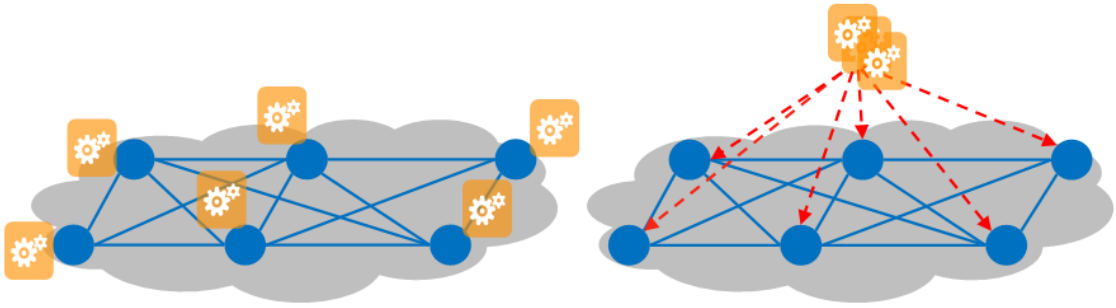

Рисунок 1.5.Відмінності в реалізації управління елементами мережі між традиційними IP-мережами та SDN.

На Рисунку 1.6 зображена загальна архітектура мережі SDN.

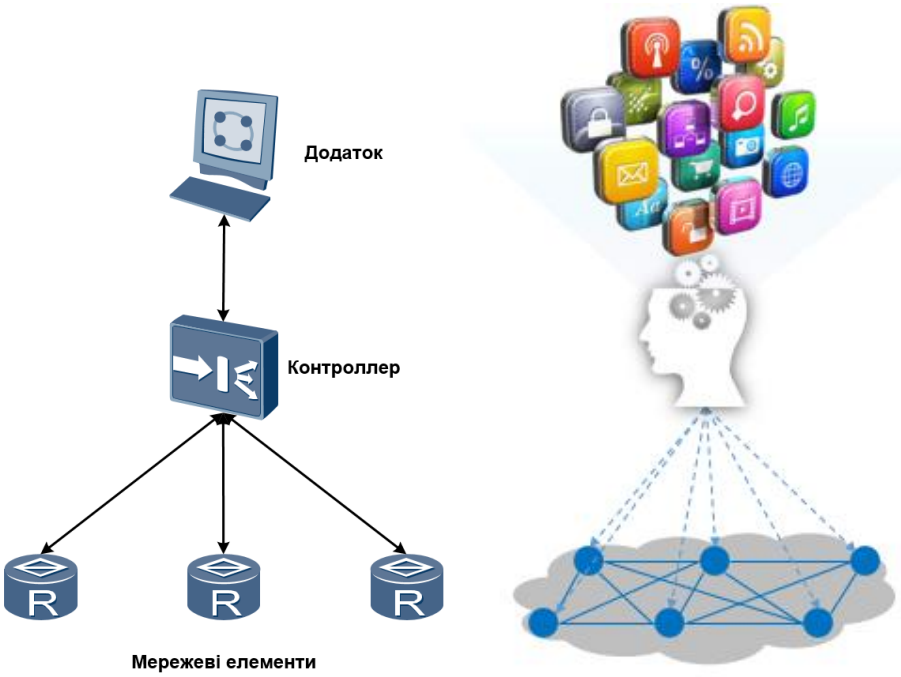

Рисунок 1.6. Загальна архітектура мережі SDN.

У мережі SDN конвергенція мережі залежить від контролера SDN, оскільки контролер SDN є основним компонентом і керує мережевими пристроями централізовано. В результаті виникає потенційна точка відмови. Проаналізувавши дану архітектуру, можна знайти 3 слабкі місця, які можуть спричинити порушення в роботі мережі SDN:

- сервер, на якому запущено контролер, може стати несправним, наприклад, сервер відключиться через проблеми з електропостачанням на місці його розташування;
- програмне забезпечення контролера може вийти з ладу;
- зв’язок між контролером та елементом мережі буде перервано.

На Рисунку 1.7 зображено варіант усунення першої слабкої точки. Він полягає у розгортанні резервного контролера. Це можна реалізувати двома способами:

- використовувати додаткові резеврні сервери. При чому вони можуть бути розміщені в різних дата-центрах;
- використовувати функцію гарячого резервного копіювання на два пристрої.

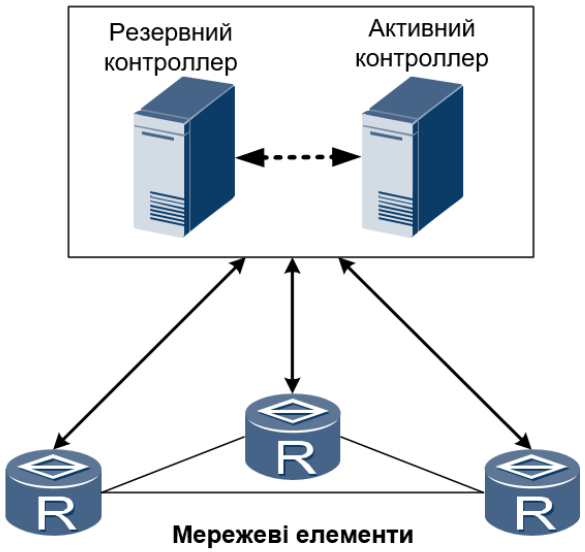

Рисунок 1.7. Варіант усунення першої слабкої точки архітектури SDN.

Функція гарячого резервного копіювання на два пристрої реалізована на основі таких механізмів:

- активний/резервний контроль, в якому для управління мережею вказані два контроллери. Для вибору активного контроллеру використовується метод пріоритетів, тобто кожному контролеру присвоєна певна величина пріоритету і активним контролером буде той, що має більше значення пріоритету;
- дані управління сервісами на активному та резервному контролерах повинні бути однаковими у сценарії гарячого резервного копіювання на два пристрої. Після того, як буде здійснено переключення активного/резервного контролерів, новий активний контролер починає функціонувати без необхідності переналаштування даних для управління сервісами.

У розподіленій архітектурі SDN є багато компонентів, які входять до її складу, такі як NETCONF, OpenFlow, PCE, RESTful, SNMP, Telnet, BGP тощо. Звичайно, кожен з них може мати певні проблеми чи помилки в роботі. Програмні компоненти мають чимало факторів, це може буде певний програмний збій або певні непередбачувані події. Такі проблеми можуть

вплинути на нормальну роботу системи. На Рисунку 1.8 зображено варіант для реалізації запобігання виникнення даних проблем.

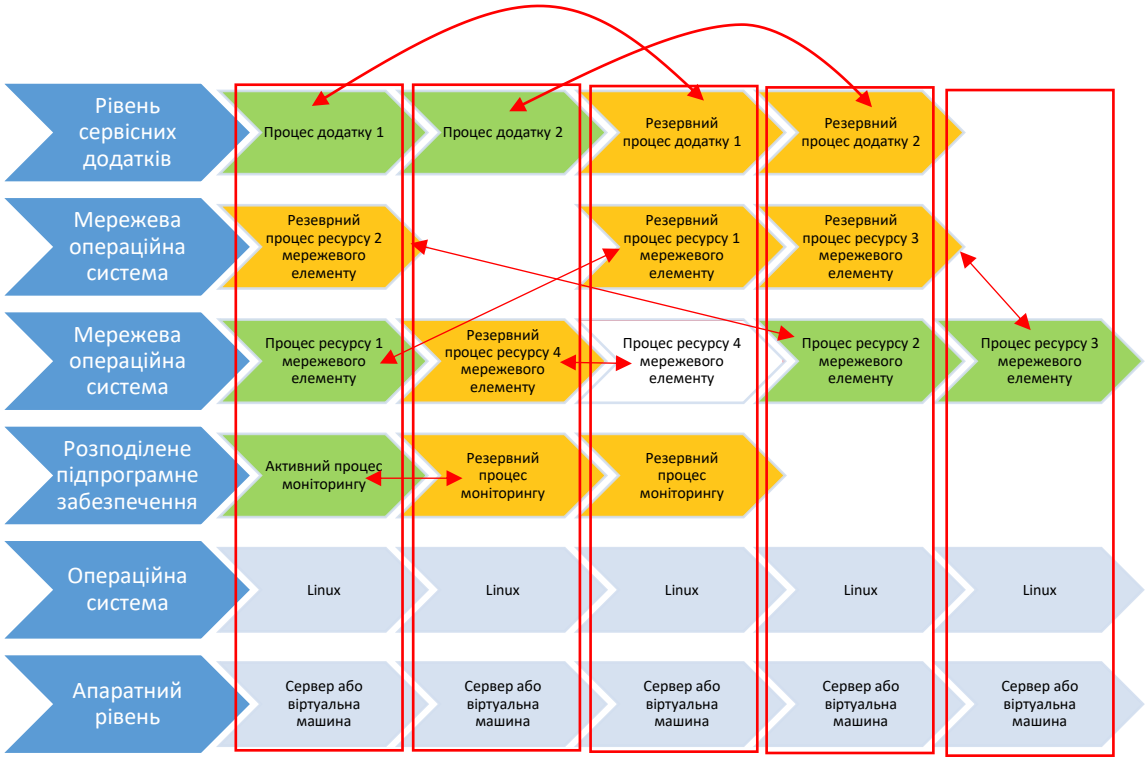

Рисунок 1.8. Варіант усунення другої слабкої точки архітектури SDN.

Кожен процес повинен мати певний механізм детектування. Без механізмів детектування на компонентах ми не в змозі виявити несправність. Наприклад, є один додатковий рівень, який називається «Розподілене підпрограмне забезпечення» і слугує для моніторингу стану програмного забезпечення, стану процесів та інших вузлів в межах кластерної системи. Для реалізації моніторингу може бути запущено кілька систем моніторингу. Крім первинної системи моніторингу, що працює на головному сервері, може бути інша система моніторингу, що працює на резервному сервері. Розподілена структура контролера може вирішити проблеми з виникненням тимчасового припинення певних процесів чи програмних збоїв.

На Рисунку 1.9 зображено варіант усунення третьої слабкої точки. Реалізувати надійний зв’язок між контроллером та елементами мережі можна шляхом додавання надлишкових незалежних каналів між контроллером та

елементами мережі або шляхом з'єднання контроллера з елементами мережі за допомогою виділеної мережі для управління.

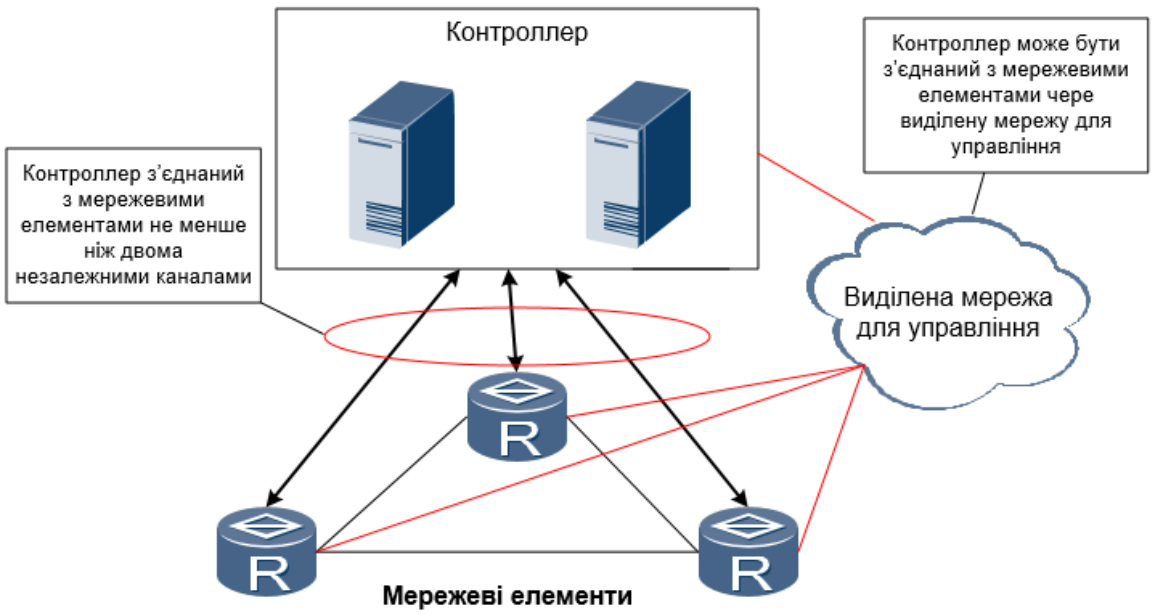

Рисунок 1.9. Варіант усунення третьої слабкої точки архітектури SDN.

Висновки до розділу 1

Головна ідея SDN - це відділення функцій передачі трафіку від функцій управління. У традиційних комутаторах і маршрутизаторах ці процеси невіддільні однин від одного. У SDN мережа, що складається з безлічі пристроїв різних виробників, постає для додатка як один логічний комутатор. SDN дозволяє адміністраторам програмувати мережу як єдине ціле, а не займатися окремими комутаторами, які можуть просто виконувати інструкції контролера.

Реалізація такої концепції значно спрощує експлуатацію мережі, її конфігурацію. Комутатори можуть бути простими і дешевими. Характеристики мережі можна оперативно змінювати в режимі реального часу. Скорочуються терміни впровадження нових додатків і сервісів. Програмні інтерфейси контролера дозволяють розробникам створювати додатки для управління мережею. Такі додатки можуть виконувати найрізноманітніші функції, причому для цього не потрібно знати особливості роботи конкретних мережевих пристроїв.

Отже, SDN дає наступні переваги:

- розподіл функцій передачі трафіку від функцій управління;
- відкриті інтерфейси між пристроями управління і передачі;
- централізоване управління мережею;
- віртуалізація фізичних ресурсів мережі;
- можливість програмування як обладнання, так і додатків;
- швидке реагування на зміни в мережі;
- можливість оптимізувати передачу трафіку (L2/L3) через більшу кількість резервних шляхів;
- можливість легше і швидше налаштовувати мережі;
- можливість істотно скоротити час розгортання додатків;
- спрощення управління мережевими пристроями;
- скорочення витрат на управління мережею;
- централізоване застосування політик, збільшення продуктивності, зменшення затримок призводить до більш ефективної взаємодії користувачів і додатків, як в корпоративних мережах, так і в мережах датацентрів;
- відкриті, засновані на стандартах протоколи дозволять взаємодіяти різним виробникам мережевого обладнання між собою, одночасно збільшуючи вибір замовнику і конкуренцію між вендорами при зниженні витрат, прискорюючи інновації як в області програмного забезпечення, так і апаратних засобів;
- контролер SDN підтримує відкритий інтерфейс програмування, який дозволяє програмувати його ззовні, створюючи середовище для автоматизації і контролю, а також масштабувати функціонал для майбутніх додатків;
- видимість всього трафіку мережі контролером.[2]

2.ПРОТОКОЛИ МЕРЕЖІ SDN

Протоколи в архітектурі SDN поділяються на 3 типи: «Північні», «Південні» та «Захід-Схід». Під «Північними» протоколами розуміються протоколи взаємодії між контролером та додатками, наприклад REST API, CORBA, SNMP, NETCONF тощо. Під «Південними» протоколами розуміються протоколи взаємодії між контролером та елементами мережі (комутаторами, маршрутизаторами), наприклад OpenFlow, NETCONF, LISP, PCEP, SNMP, OVSDB тощо. Під протоколами типу «Захід-Схід» розуміються протоколи взаємодії елементів одного рівня між собою, наприклад BGP.

В даному розділі детально описано принципи дії, особливості, структуру та зроблено порівняння двох найбільш поширених «Південних» протоколів: OpenFlow та NETCONF.

2.1.Протокол OpenFlow

Протокол OpenFlow - це перший стандартизований протокол «Південного» інтерфейсу, який використовується в стандартній мережевій архітектурі SDN.

2.1.1.Базова архітектура протоколу OpenFlow

Однією з важливих характеристик архітектури SDN є розділення площини управління та площини передачі даних. У відповідності до цих характеристик, в архітектурі OpenFlow були введені дві ролі для апаратної частини – це OpenFlow контролер та OpenFlow комутатор.

Контролер OpenFlow служить блоком управління для обчислення маршрутів та розподілу даних до OpenFlow комутаторів, які відповідають за переадресацію пакетів на основі отриманих записів потоків.

OpenFlow комутатор - це пристрій, який отримує команди або інформацію про таблиці потоків від контролера та відправляє на контролер інформацію про їх стан. На основі інформації про потоки, що генерується та

надсилається з контролера, OpenFlow комутатор функціонує як пристрій для фізичного пересилання пакетів даних.

OpenFlow комутатор складається з (Рисунок 2.1):

- таблиць потоків;
- таблиці груп;
- захищеного каналу.

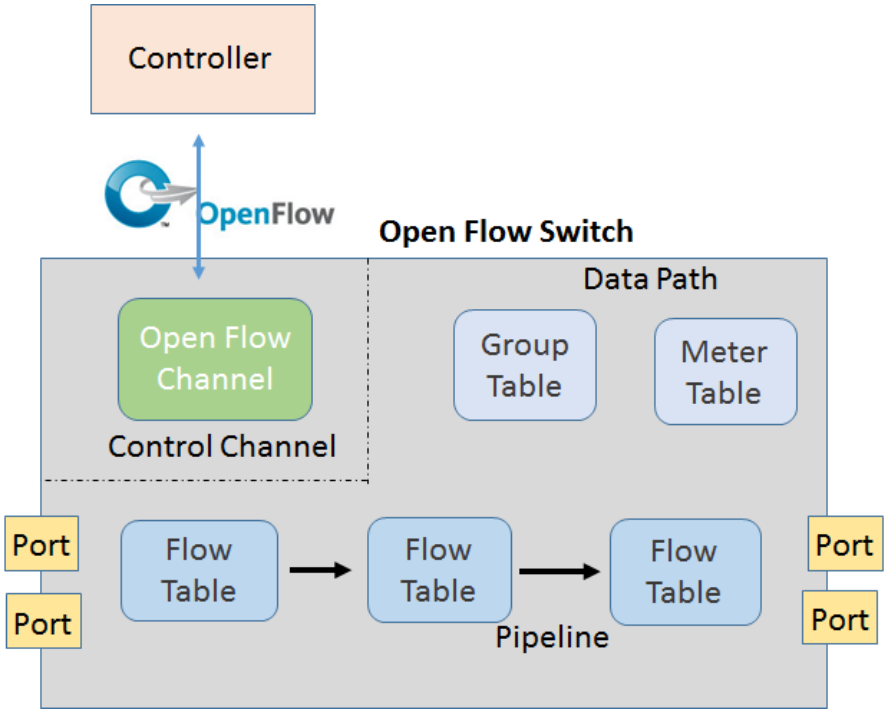

Рисунок 2.1. Складові OpenFlow комутатора.

Таблиця потоків в комутаторі OpenFlow складається із записів, які містять інформацію про поля відповідності, лічильники та набори інструкцій, які слід застосувати для відповідних пакетів.

У комутаторі OpenFlow можуть бути кілька таблиць потоків, які будуть опрацьовані за допомогою конвеєрної обробки. Наприклад, коли пакет приходить на комутатор і відповідає запису потоку в першій таблиці потоків, відповідна інструкція буде пов'язана з інструкцією конвеєрної обробки, яка дозволяє передавати пакети в наступні таблиці для подальшої обробки. Конвеєрна обробка таблиці припиняється, якщо інструкції, пов'язані з відповідним записом потоку, не визначають наступну таблицю потоків.

Таблиця груп містить список записів групи, і кожен запис групи асоціюється з пакетами дій або групами дій. Пакети, які визначені у групі та збігаються із записами групи, будуть опрацьовані відповідно до одного або кількох пакетів дій.

Захищений канал використовується для підключення до зовнішнього контролера та встановлює канал зв'язку через протокол OpenFlow.

OpenFlow комутатор повинен підтримувати 3 типи стандартних портів OpenFlow:

- фізичні порти, які слугують для з’єднання комутаторів між собою та для підключення до зовнішньої мережі;
- логічні порти, які є портами вищого рівня і не асоціюються напряму з фізичними портами, та можуть бути реалізовані за допомогою таких технологій, як агрегація портів, тунельні інтерфейси чи loopback-інтерфейси. Комутатори OpenFlow підключаються один до одного логічно через порти OpenFlow. Пакети OpenFlow поступають на вхідний порт, далі пакети проходять через конвеєрну обробку, і можуть бути передані на вихідний порт.
- резервні порти є визначеними специфікацією OpenFlow портами, які використовуються для певних дій по переадресації після того, як було знайдено відповідніть в записах потоків.

Нижче перелічені обов'язкові зарезервовані порти:

- ALL: означає, що кожен порт на комутаторі може використовуватися для переадресації пакетів та працює як вихідний порт;
- CONTROLLER: представляє порт, який використовується для каналу зв'язку між комутатором і контролером; він може працювати як вхідний або вихідний порт;
- TABLE: представляє початок конвеєрної обробки OpenFlow, щоб подати отримані пакети до першої таблиці потоків для конвеєрної обробки. Він може працювати як вихідний порт;

- IN_PORT: представляє вхідний порт для пакетів;
- ANY: спеціальний порт, де не буде здійснюватися жодна дія по обробці. Він не може бути ні вхідним, ні вихідним портом.

Крім необхідних зарезервованих портів, є деякі додаткові порти, які описано нижче:

- LOCAL: представляє внутрішні порти комутатора, що використовуються для управління комутатором. Він може використовуватися для здійснення підключення до контролера через мережу OpenFlow і може працювати як вхідний або вихідний порт;
- NORMAL: представляє традиційний не OpenFlow конвеєр комутатора, який використовується для передачі пакету з конверєної обробки OpenFlow до нормальної конвеєрної обробки і може працювати як вихідний порт;
- FLOOD: представляє порт, який може поширити всі пакети на всі інші стандартні порти, крім вхідних портів.

Залежно від резервних портів, які підтримуються комутаторами, їх можна класифікувати на OpenFlow-Only комутатор, який не підтримує функції NORMAL і FLOOD та OpenFlow-Hybrid комутатор, який підтримує всі функції портів. Іншими словами, OpenFlow-Hybrid комутатор сумісний як з елементами мережі OpenFlow, так і з елементами традиційної мережі Ethernet.

З виходом кожної нової версії протоколу OpenFlow, зростає різноманітність функціоналу, який вона підтримує. Це призводить до часткової несумісності контролера та мережевого обладнання, яке використовується. З ціллю вирішення даної проблеми, співтовариством ONF був запропонований підхід під назвою OpenFlow Table Type Patterns. Суть даного підходу полягає в тому, як можна досягти максимальної сумісності функції, доступних в новій версії OpenFlow та функцій, які підтримує мікросхема ASIC мережевого елементу.

Основним завданням мережевого елементу є обробка вхідних пакетів за вказаними контролером правилами та перенаправлення пакетів на вихідні інтерфейси. Під обробкою пакетів мається на увазі безліч операцій, які повинні бути визначені в мікросхемі ASIC мережевого елементу для підтримки певного пайплайну. Під пайплайном мається на увазі послідовність операцій, яку виконує мережевий елемент над пакетом до етапу його перенаправлення на вихідний інтерфейс. На Рисунку 2.2 зображено приклад пайплайну для реалізації маршрутизації та мережевого мосту.

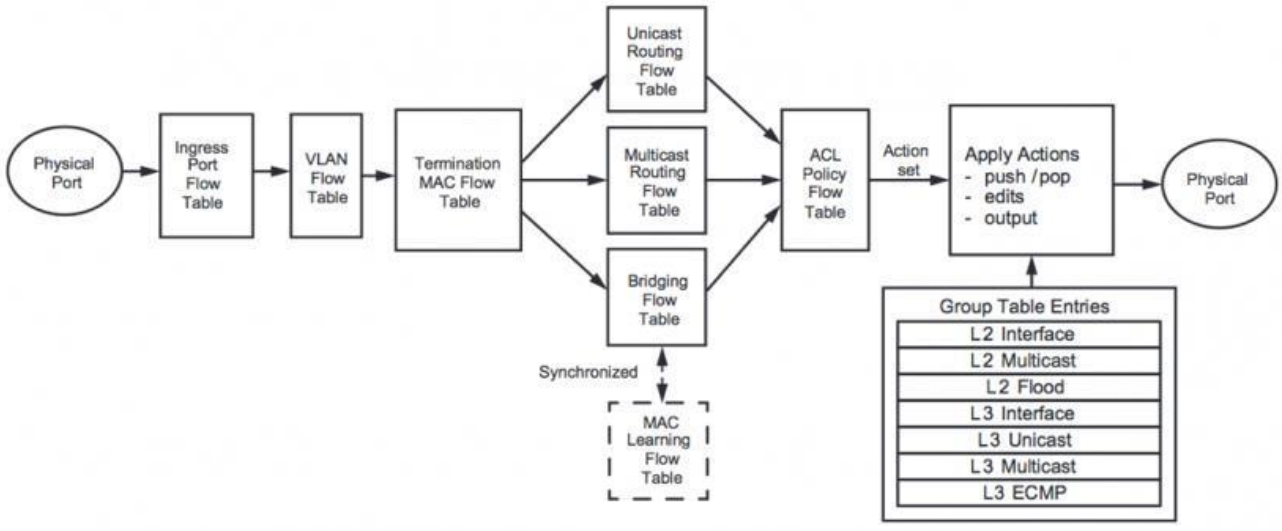

Рисунок 2.2. Пайплайн для реалізації маршрутизації та мережевого мосту.

Контролер OpenFlow генерує багато пайплайнів, які виконують різні операції над пакетами і в результаті може виникнути несумісніть з пайплайнами, які закладені в мікросхемі ASIC. Тобто певні інструкції агент OpenFlow, який знаходиться в комутаторі, не зможе виконати, якщо ASIC не підтримує їх. Дані розбіжності повинні бути виявлені на початковому етапі взаємодії контролера та комутатора. Необхідно порівнювати пайплайни OpenFlow з функціоналом мікросхеми ASIC, але контролер SDN ще немає механізму для реалізації даної функції, що є недоліком.

Варіантом вирішення даної проблеми є використання обмежених пайплайнів, які підтримують обидві сторони: контролер та комутатор. Даний варіант отримав назву TTP. TTP є файлом у форматі JSON, який містить в собі інформацію про типи повідомлень, кількість таблиць та перелік операцій, які

будуть виконуватись над пакетом. Для запобігання виникнення ситуацій, коли комутатор отримує незрозумілі для нього вказівки контролера, необхідно спочатку визначити TTP, який підтримується обома сторонами. Для цього розбробники SDN та виробники мережевого обладнання повинні узгодити підтримувані TTP. Процес узгодження відбувається наступним чином:

1. виробник обладання або розробник SDN з ціллю отримання підтримки логіки роботи свого продукту розробляє певний TTP;
2. виробник обладання або розробник SDN ділиться даним TTP з партнерами;
3. виробник обладання або розробник SDN додають підтримку даного TTP до своєї продукції;

Після завершення даних етапів, обидва пристрої можуть працювати в рамках однакових TTP, без виникнення несумісностей між ними.

2.1.2.Таблиці протоколу OpenFlow

Починаючи з OpenFlow версії 1.1 і вище, протокол OpenFlow підтримує конвеєрну обробку у поєднанні з можливостями підтримки декількох таблиць потоків. Порівняно з OpenFlow версії 1.0, яка підтримує одну таблицю потоків, конвеєрна обробка зменшує розмір для записів відповідності в таблиці та покращує ефективність перевірки таблиці потоків, розділяючи поля відповідності таблиці потоків на кілька таблиць потоків.

Таблиця потоків для комутатора OpenFlow послідовно пронумерована, починаючи з 0. Коли пакет передається в першу таблицю потоків, то обирається відповідність з найвищим пріоритетом, а набір інструкцій, який їй відповідає, спрямовуватиме пакет до наступної таблиці потоків, і процес обробки пакету повториться знову (Рисунок 2.3). Якщо інструкція не спрямовує пакет до наступної таблиці, конвеєрна обробка зупиняється і над пакетом буде виконана відповідна дія, якщо вона встановлена.

Якщо пакет не відповідає жодному запису потоку в таблиці потоків, то це називається пропуском таблиці. Якщо для запису пропуску таблиці зроблена особлива конфігурація, запис потоку пропуску таблиці можна обробляти

різними наборами дій, такими як відкидання, прозоре пересилання їх до контролера тощо.

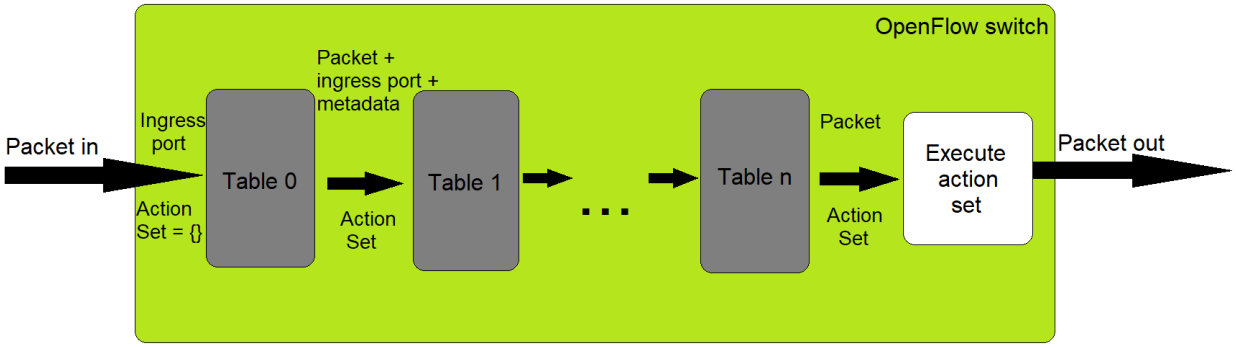

Рисунок 2.3. Потік пакетів через конвеєрну обробку OpenFlow.

Після отримання пакету комутатор OpenFlow починає його обробку, перевіряючи першу таблицю потоків. На основі інформації про поля відповідності в пакетах, таких як MAC-адреса, IP-адреса або поля метаданих, лічильники таблиці потоків оновлюються та виконуються відповідні інструкції, такі як виконання або оновлення набору інструкцій, оновлення полів відповідності пакетів або оновлення метаданих для обробки наступної потокової таблиці (Рисунок 2.4).

Якщо є метадані, визначені для обробки наступною таблицею потоків, то пакет буде перенаправлений до наступної таблиці потоків, повторно виконуючи перевірку полів відповідності та перевірку інструкцій. Якщо пакети вже були надіслані до останньої таблиці потоків з найбільшим порядковим номером, це означає, що більше немає таблиці потоків, яку потрібно перевірити, і в такому випадку буде виконано набір дій, визначений в останній таблиці потоків.

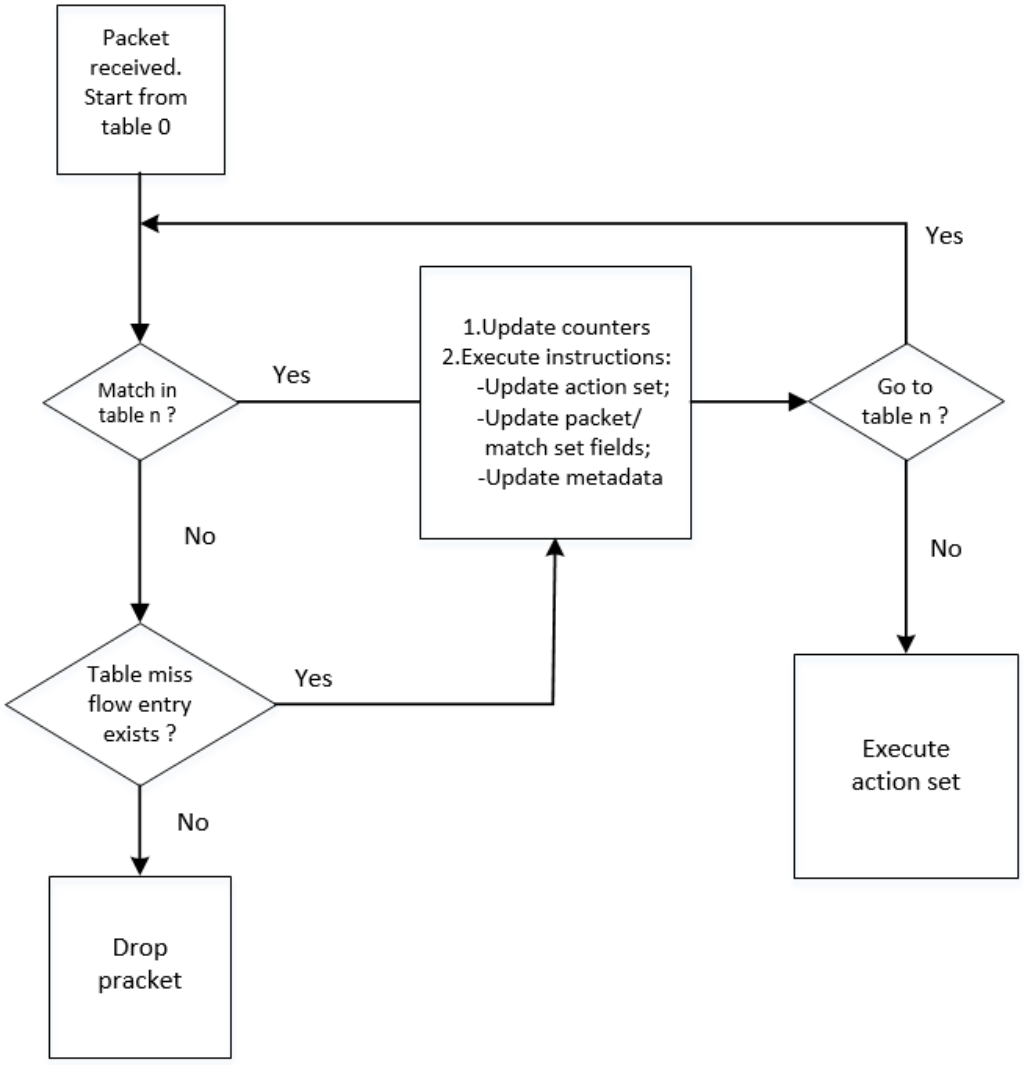

Рисунок 2.4. Алгоритм обробки пакетів в таблицях потоків OpenFlow.

У випадку коли в таблиці потоків немає відповідних записів потоку, пакет буде відкинуто, якщо на комутаторі не буде налаштовано запис пропуску таблиці. Якщо використовується функція пропуску таблиці, то пакет буде оброблятися згідно інструкцій, визначених для випадку пропуску таблиці.

Протокол OpenFlow1.4 визначає 3 типи таблиць: таблицю потоків, таблицю груп та таблицю виміру.

2.1.2.1.Таблиця потоків

Таблиця потоків є основним елементом протоколу OpenFlow. Кожен запис таблиці потоків містить наступні елементи (Рисунок 2.5):

- match fields, що складається з вхідних портів та необов'язкових відповідних полів або метаданих, зазначених у попередній таблиці;
- priority, яка визначає пріоритет запису таблиці потоків;
- counters, який показує кількість пакетів, що відповідають даному запису і оновлюється в режимі реального часу;
- instructions,в яких зазначено зміни наборів дій чи конвеєрної обробки;
- timeouts, який показує максимальний час життя потоку;
- cookie, який показує вибране контролером значення, яке, використовується ним для фільтрації потоку, модифікації потоку або видалення потоку. Даний елемент не використовується під час обробки пакетів.

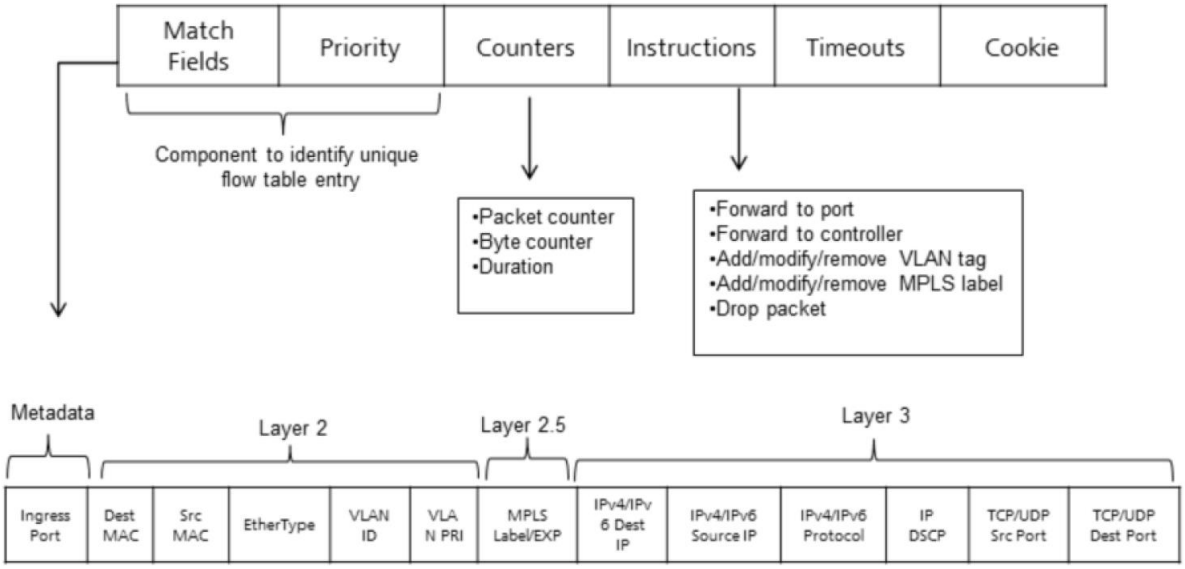

Рисунок 2.5.Формат таблиці потоків OpenFlow.

Поля match fields та priority використовуються для ідентифікації відповідних записів таблиці потоків.

В якості значення поля match fields для протоколу OpenFlow 1.4 використовуються:

- вхідний порт;
- MAC-адреса призначення;
- MAC-адреса джерела;
- тип Ethernet;
- номер протоколуIPv4/IPv6;

- IPv4/IPv6 адреса джерела;
- IPv4/IPv6 адреса призначення;
- TCP/UDP порт джерела;
- TCP/UDP порт призначення.

Після того, як буде знайдена відповідність пакету та запису потоку з найвищим пріоритетом, пакети будуть оброблятися на основі відповідних інструкцій, таких як модифікація пакетів, застосування наборів дій або виконання конвеєрної обробки.

Нижче показано приклади інструкцій:

- “Meter” - направити пакети до певного лічильника для виконання певних операцій, таких як фільтрація, відкидання або зміна значення DSCP в заголовку IP-пакета;
- “Apply-action» - негайно застосувати конкретні дії без будь-яких їх змін;
- “Clear-action” - негайно видалити всі дії в наборі;
- “Write-actions” - негайно додати або модифікувати набір дій;
- “Goto-Table” - направити пакети для обробки наступною таблицею.

Незалежно від порядку внесення певних дій до набору дій, їх пріоритет в наборі дій виглядає наступним чином:

- “Copy TTL inwards” – копіювати TTL із зовнішнього заголовка;
- “Pop” - видалити всі теги;
- “Push-MPLS” – додати тег MPLS;
- “Push-PBB” - додати тег PBB;
- “Push-VLAN” - додати VLAN-тег;
- “Copy TTL outwards” - копіювати TTL із внутрішнього заголовка;
- “Decrement TTL” – з меншити TTL;
- “Set” - застосувати всі “set-fields” дії до пакетів;
- “QoS” - застосувати всі дії QoS;
- “Group” - застосувати групу дії до пакетів;
- “Output” - переслати пакет на вказаний порт.

Таблиці потоків підтримують також запис пропуску таблиці, щоб обробляти пакети, які не відповідають жодному запису з таблиці потоків. Запис пропуску таблиці має найнижчий пріоритет в таблиці потоків і не має поля “Match-filed” в записі таблиці потоків. За стандартними налаштуваннями, функція пропуску таблиці відключена і всі пакети, які не відповідають жодному запису з таблиці потоків, будуть відкидатись. Дану функцію можна активувати в налаштуваннях комутатора. Записи пропуску таблиці пітримують багато різних дій, наприклад відправлення пакетів до контролера через резервний порт, відкидання пакетів за допомогою дії “Clear-action”, або направлення пакетів до наступної таблиці потоків.

Коли видаляється запис з таблиці потоку, то комутатор надсилає повідомлення про видалення потоку на контролер, яке містить повний опис запису потоку, причину видалення потоку, тривалість даного запису потоку, та його статистику.

Запис таблиці потоків може бути видалений із таблиці потоку з наступних причин:

1. По запиту контролера:
   - контролер може видаляти записи потоку, надсилаючи повідомлення «DELETE» на комутатор OpenFlow;
   - видалення контролером запису групи або запису виміру може також призвести до видалення запису потоку.
2. Закінчення терміну дії запису потоків:
   - механізм закінчення терміну дії запису потоків запускається на комутаторі OpenFlow та працює незалежно від контролера;
   - існує два типи таймерів, пов’язаних із записом таблиці потоків:
     i. Idle_timeout – вказує інтервал часу після закінчення якого необхідно видалити запис потоків, якщо протягом визначеної тривалості idle_timeout немає пакетів, що відповідають параметрам даного запису;

ii. Hard_timeout - вказує тривалість примусового видалення запису потоків незалежно від того, чи є пакети, що відповідають параметрам даного запису.

3. Механізм видалення записів комутатором - це додаткова функція, яка за замовчуванням не включена. Після того, як механізм видалення записів комутатором увімкнено, запис таблиці потоків може бути видалений, коли комутатору необхідно перерозподілити ресурси на основі статистичних даних, таких як параметри взапису потоку, планування ресурсів, тощо.

Приклад таблиці потоків зображено на Рисунку 2.6.

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
•	10:20:.	•	•	•	•	port 1	250
•	•	•	5.6.7.8	•	•	port 2	300
•	•	•	•	25	•	drop	892
•	•	•	192.	•	•	local	120
•	•	•	•	•	•	controller	11

Рисунок 2.6. Типова таблиця потоків в мережевому пристрої, який підтримує OpenFlow.

2.1.2.2.Таблиця груп

Основними компонентами запису групи в таблиці груп є (Рисунок 2.7):

- “Group Indentifier” – унікальне 32-бітне число, яке слугує для ідентифікації групи;
- “Group Type” – визначає певний тип групи;
- “Counters” – дана величина зібльшується на 1, коли пакет обробляється даною групою;
- “Actions buckets” – набір дій, який виконується відповідно до вказаних параметрів.

Group Identifier	Group Type	Counters	Action Buckets

Рисунок 2.7. Формат таблиці груп.

Комутатор OpenFlow повинен підтримувати обов'язкові типи груп, описані нижче:

- ALL - ця група використовується в основному для передачі в режимі багатоадресної або широкомовної передачі; пакети реплікуються та обробляються всіма наборами дій певної групи групи;
- INDIRECT - ця група підтримує лише один набір дій, що дозволяє обробляти пакети лише одним визначеним набором дій в цій групі.

Інші необов'язкові типи груп включають:

- SELECT - пакети в цій групі будуть оброблятись одним обраним набором в групі на основі алгоритму вибору комутатора, такого як хеш-алгоритм тощо;
- FAST FAILOVER - ця група включає набори дій, пов'язані з доступністю портів. Пакети обробляються першим доступним набором, щоб досягти швидкого пересилання пакетів на основі доступних портів. Якщо немає доступних портів, то не буде й доступних наборів дій, тоді пакети будуть відкидатись.

2.1.2.3.Таблиця виміру

Основними компонентами таблиці виміру є (Рисунок 2.8):

- “Meter Identifier” – унікальне 32-бітне число для ідентифікації виміру;
- “Meter Bands” – кожна таблиця виміру може мати одну або більше величину смуги. Кожне значення смуги пропускання ідентифікується типом смуги, яка визначається типом дії, наприклад відкидання пакетів чи зміна значення DSCP. Основні компоненти “Meter Bands” зображені на Рисунку 2.9.

- “Counters” – лічильник опрацьованих пакетів.

Meter Identifier	Meter Bands	Counters

Рисунок 2.8. Формат таблиці виміру.

Band Type	Rate	Counter	Arguments

Рисунок 2.9. Основні компоненти “Meter Bands”.

Таблиця виміру дозволяє OpenFlow виконувати прості операції QoS, такі як зміна Diffserv, обмеження смуги пропускання, тощо.

2.1.3.Типи повідомлень протоколу OpenFlow

Структура повідомлень OpenFlow зображена на Рисунку 2.10.

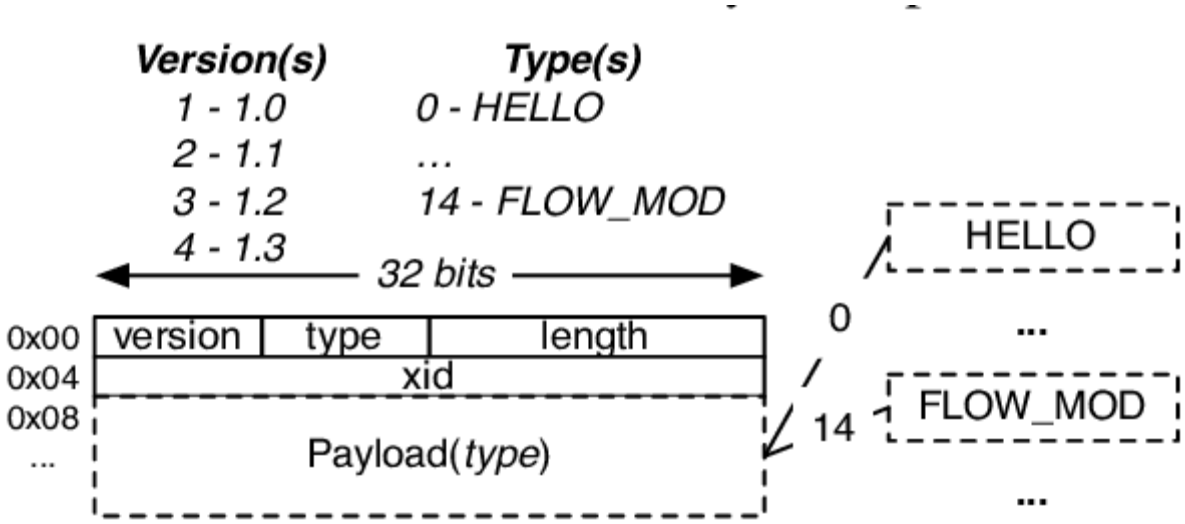

Рисунок 2.10. Структура повідомлень OpenFlow.

Поле «version» вказує версію протоколу OpenFlow, якій належить це повідомлення. Поле «type» вказує, який тип повідомлення передається. Поле «length» вказує де це повідомлення закінчиться в потоці байтів, починаючи з першого байту заголовка. Ідентифікатор «xid» або ідентифікатор транзакції використовується для співставлення запитів та відповідей.

Згідно з документацією, протоколи OpenFlow поділяються на 3 типи:

- від контролера до комутатора, які ініціюються контролером, в основному для запиту певної інформації або основних можливостей і характеристик від комутатора, що можуть знадобитися під час встановлення каналу OpenFlow. Прикладами даного типу повідомлень є повідомлення Modify-State, які використовуються для додавання, видалення або зміни записів потоку в таблицях потоків; повідомлення Read-State, яке використовується для збору інформації, такої як дані конфігурації від комутатора; повідомлення Packet-Out, яке використовується для пересилання пакетів, отриманих через Packet-In повідомлення тощо;
- асинхронні повідомлення, які надсилаються контролеру комутатором, щоб підтвердити отримання пакета або сповістити про зміну стану порта на комутаторі. Прикладами цього типу повідомлень є повідомлення Packet-In, яке використовується для передачі пакета до контролера; повідомлення Flow-Removed, яке використовується для інформування про видалення певного запису в таблиці потоків, і повідомлення Port-Status, яке використовується для повідомлення про зміну стану порта на комутаторі;
- симетричні повідомлення, які можуть бути надіслані від комутатора до контролера і навпаки. Прикладами симетричних повідомлень є повідомлення Hello, які використовуються під час початку встановлення з'єднання, повідомлення Echo request/reply, які використовуються для перевірки стану з'єднання та повідомлення сусіднього елементи про наявні помилки.

2.2.Протокол NETCONF

NETCONF – це протокол для конфігурації та управління мережею, який базується на мові розмітки XML. Даний протокол був розроблений та стандартизований організацією IETF. NETCONF використовує кодовані XML виклики віддалених процедур (RPC), та дані конфігурації для управління мережевими пристроями. Обов'язковим транспортним протоколом для NETCONF є протокол SSH. За замовчуванням він використовує TCP-порт 830.

Сервер NETCONF повинен прослуховувати на даному порті з'єднання з підсистемами NETCONF.

NETCONF пропонує наступні переваги:

- спрощує керування даними конфігурації та сумісність між пристроями різних виробників, використовуючи кодування XML та механізм RPC для зміни даних конфігурації;
- зменшує ймовірність виникнення аварій на мережі, які зазвичай спричиняються помилками при налаштуваннях вручну, тобто спричинених людським фактором;
- покращує ефективність оновлення програмного забезпечення системи, що здійснюється за допомогою використання конфігураційних інструментів;
- забезпечує високу здатність до розширення, дозволяючи різним постачальникам визначати додаткові операції протоколу NETCONF;
- покращує захист даних за допомогою механізмів аутентифікації та авторизації;

Протокол NETCONF характеризується наявністю мови моделювання даних YANG, яка представляє структуру даних у форматі дерева XML. Як показано на Рисунку 2.11, протокол NETCONF дозволяє програмам управління переглядати, маніпулювати або змінювати дані в моделях YANG, а модель YANG в свою чергу служить моделлю даних NETCONF, яка використовується для точного визначення структури, синтаксису та семантики даних.

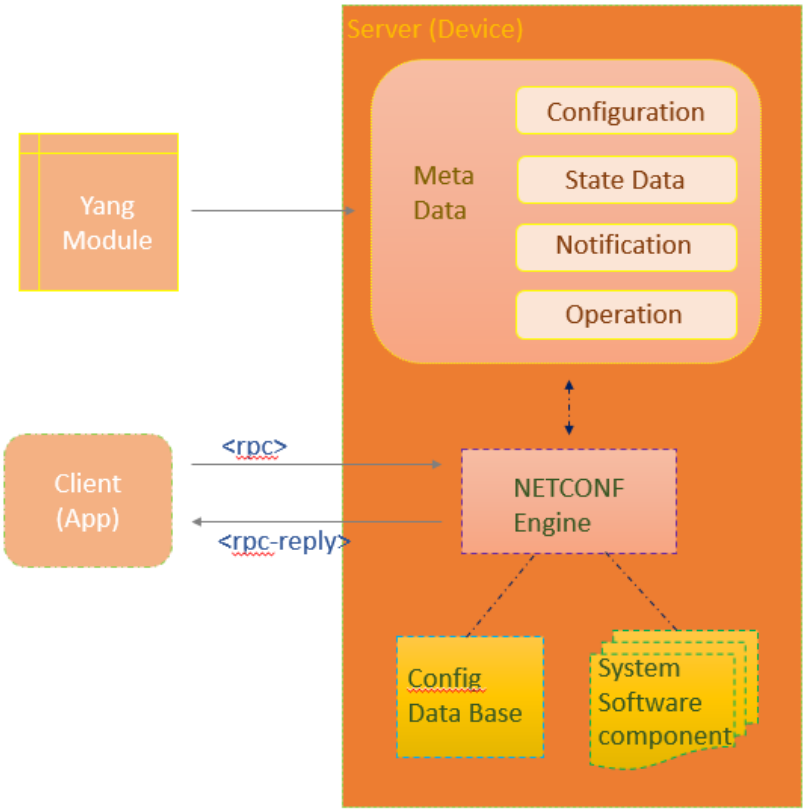

Рисунок 2.11. Структура протоколу NETCONF на базі моделі YANG.

Модуль YANG визначає ієрархію даних, які можуть бути використані для операцій протоколу NETCONF, включаючи конфігурацію, дані про стан, RPC та повідомлення. Це дозволяє повністю описати всі дані, що пересилаються між клієнтом NETCONF та сервером. Мережеві пристрої мають компонент системного програмного забезпечення (System Software Component), який відповідає за виконання мережевих операцій. Він працює на основі інформації, яка зберігається в конфігураційній базі даних. Пристрій, що підтримує NETCONF, складається з елементу NETCONF Engine та стеку протоколів, які дозволяють клієнту NETCONF взаємодіяти з пристроєм та працювати з даними, які він містить. Клієнт NETCONF може ініціювати надійний RPC-зв’язок для взаємодії з елементом NETCONF Engine, що працює на пристрої. Це з'єднання дозволяє клієнту працювати з наборами даних певного пристрою. Інформація, що міститься в метаданих, класифікується за категоріями та призначенням. Дана інформація моделюється та зберігається у вигляді набору даних YANG. Системне програмне забезпечення відповідає за оновлення змін у конфігураційній базі даних для активації сервісів.

Мережева архітектура NETCONF складається з таких компонентів (Рисунок 2.12):

1. NETCONF менеджер, який:
    - функціонує як клієнт, який використовує протокол NETCONF для управління пристроями;
    - може надсилати RPC-повідомлення агенту NETCONF керованого пристрою для запиту чи зміни його конфігураційних даних;
    - може дізнатися стан керованого пристрою на основі аварій та подій, про які його повідомляє агент NETCONF.
2. NETCONF агент, який:
    - функціонує як сервер, що підтримує конфігураційні дані на керованому пристрої, відповідає на RPC-повідомлення, що надсилаються менеджером NETCONF, і надсилає запитувану інформацію менеджеру NETCONF. Після того, як агент NETCONF отримує RPC-повідомлення, він його аналізує, обробляє на основі CMF і відправляє повідомлення RPC-Reply менеджеру NETCONF;
    - повідомляє про аварії або події менеджеру NETCONF, щоб він знав про стан керованого пристрою.
3. Schema-файл, який:
    - є файлом моделі даних, що визначає набір правил, які використовуються для опису документа XML. Schema-файл визначає всі об'єкти управління на керованому пристрої, обмеження та ієрархічні зв’язки між цими об'єктами управління, а також дозволи на читання та запис цих об'єктів управління;
    - функціонує за принципом, схожим до принципу роботи файлу MIB протоколу SNMP.

Інформація, яку можна отримати із мережевого пристрою, така:

- конфігураційні дані: набір даних, що записуються, і які необхідні для перетворення пристрою з його початкового стану за замовчуванням у його поточний стан;

- додаткові дані: додаткові неконфігураційні дані на пристрої, такі як інформація для читання та зібрана статистика.

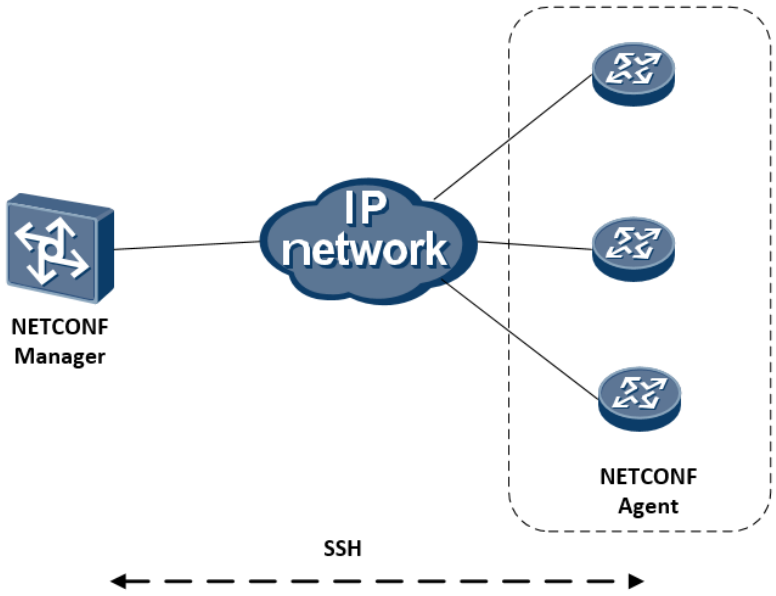

Рисунок 2.12. Мережева архітектура протоколу NETCONF.

NETCONF лише займається операціями з конфігураційними даними, які виконуються менеджером NETCONF, і не пов'язаний з тим, як ці дані конфігурації зберігаються.

2.2.1.Рівні протоколу NETCONF

Фреймворк протоколу NETCONF має ієрархічну структуру (Рисунок 2.13). Як і в моделі OSI, нижній рівень надає послуги верхньому рівню. Ієрархічна структура дозволяє кожному рівню зосередитись лише на одному аспекті NETCONF і зменшує взаємозалежність між різними рівнями.

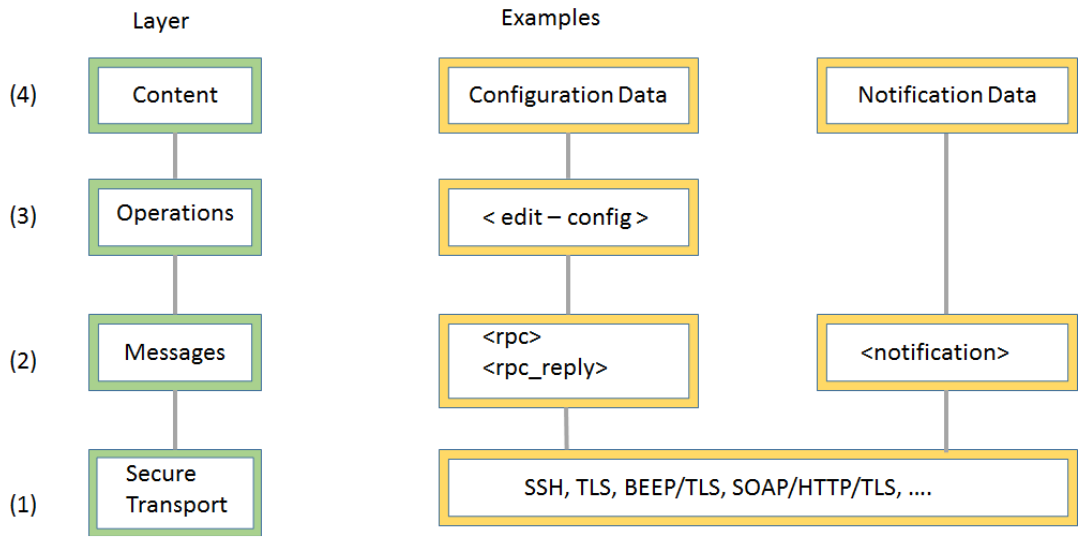

Рисунок 2.13. Рівні протоколу NETCONF.

Структура протоколу NETCONF поділяється на наступні рівні:

1. захищений транспортний протокол: NETCONF може працювати поверх будь-якого транспортного протоколу, який відповідає наступним вимогам:
   - транспортний протокол є орієнтованим на з'єднання. Тобто між менеджером NETCONF та агентом встановлюється постійний зв’язок, який забезпечує надійний та послідовний обмін даними;
   - транспортний протокол забезпечує аутентифікацію, цілісність даних та конфіденційність для NETCONF;
   - транспортний протокол забезпечує механізм розпізнавання типу сесії (клієнта чи сервера) для NETCONF;
2. рівень повідомлень забезпечує простий і незалежний від транспорту механізм фреймування для кодованих RPC. Менеджер NETCONF використовує елемент <rpc> для інкапсуляції інформації повідомлень RPC-Reply та передає її агенту NETCONF через захищену та орієнтовану на з'єднання сесію. Агент NETCONF використовує елемент <rpc-reply>, щоб інкапсулювати інформацію повідомлень RPC-Reply (дані рівнів операцій та вмісту) та надсилає інформацію повідомлень RPC-Reply менеджеру NETCONF. Зазвичай елемент <rpc-reply> інкапсулює дані, необхідні менеджеру NETCONF, або інформацію про конфігураційні дані. Якщо менеджер NETCONF надсилає неправильний запит або агенту

NETCONF не вдається обробити запит від менеджера NETCONF, агент NETCONF інкапсулює елемент <rpc-error>, що містить детальну інформацію про помилки в елементі <rpc-request>, і надсилає <rpc-reply> менеджеру NETCONF;

3. рівень операцій визначає перелік базових операцій, які використовуються в RPC. Базові операції складаються з основних можливостей протоколу NETCONF;
4. рівень вмісту описує конфігураційні дані, що беруть участь в управлінні мережею. Дані конфігурації залежать від пристроїв конкретного виробника. Усі рівні є стандартизованими для NETCONF, окрім рівня вмісту. У рівні вмісту немає стандартної мови моделювання даних NETCONF або моделі даних.

Протокол NETCONF використовує модель зв'язку, що базується на повідомленнях RPC. В Таблиці 2.1 містяться найбільш поширені елементи RPC.

Таблиця 2.1. Елементи RPC.

Елемент	Опис
<rpc>	Запит, що NETCONF менеджер надсилає агенту
<rpc-reply>	Відповідь на <rpc> повідомлення, яке агент відправляє менеджеру
<rpc-error>	Інкапсульований в <rpc-reply> елемент агентом NETCONF, щоб сповістити менеджера про помилку в обробці повідомлення <rpc>
<ok>	Відправляється агентом менеджеру, щоб сповістити про те, що протягом обробки повідомлення <rpc> не виникло помилок

2.2.2.Можливості протоколу NETCONF

На першому етапі ініціації NETCONF-сесії, кожен її учасник надсилає повідомлення Hello (Рисунок 2.14), що містить перелік його можливостей. Якщо обидва елементи підтримують дані можливості, то вони можуть реалізувати спеціальні функції управління, що базуються на цих можливостях. Кожен учасник NETCONF-сесії надсилає повідомлення Hello, як тільки з'єднання починається і не чекає отримання повідомлення з іншого боку. Після того, як агент NETCONF обмінявся Hello-повідомленнями з менеджером NETCONF - агент чекає на <rpc> елемент від менеджера NETCONF. У відповідь на кожен <rpc> агент NETCONF відправляє <rpc-reply>. Hello-повідомлення, що надсилаються агентами NETCONF, можуть відрізнятись в різних виробників.

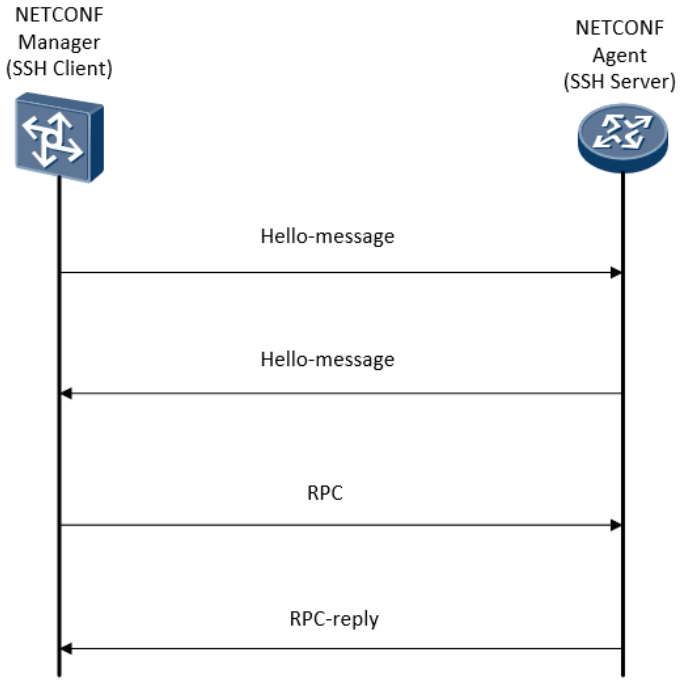

Рисунок 2.14. Процес ініціалізації NETCONF-сесії.

На Рисунку 2.15 зображено приклад вмісту XML-документу з переліком можливостей, який передається в Hello-повідомленні менеджером NETCONF.

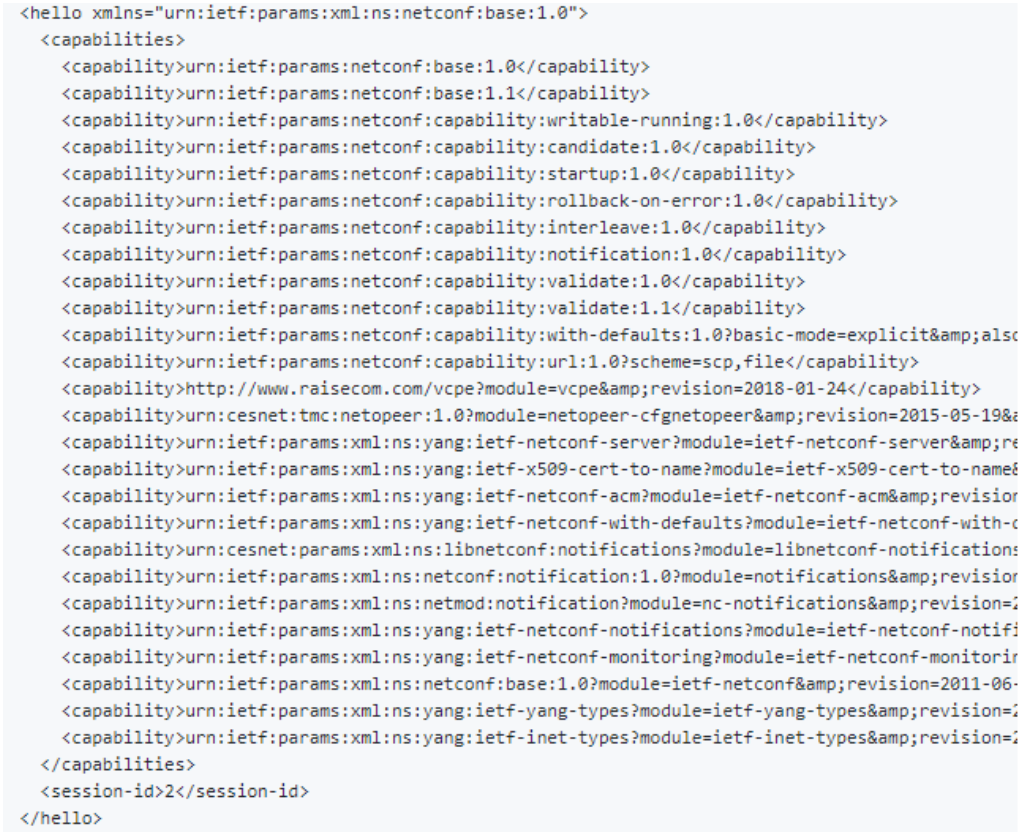

Рисунок 2.15. XML-документ з переліком можливостей, який передається в Hello-повідомленні менеджером NETCONF.

Базові операції та можливості протоколу NETCONF містяться в Таблиці 2.2.

Таблиця 2.2. Базові операції та можливості протоколу NETCONF.

Операція	Опис
<get>	Отримати інформацію про поточну конфігурацію та про поточний стан пристрою з конфігураційної бази даних <running/>
<get-config>	Отримати інформацію про всю або конкретну конфігурацію з конфігураційних баз даних <running/>, <candidate/> та <startup/>. Також дана операція дозволяє отримати конфігурацію з файлу <url>config.cfg</url>.
<edit-config>	Змінювати, створювати, перезаписувати або видаляти дані конфігурації

Операція	Опис
<copy-config>	Копіювати всю конфігураційну базу даних до іншої бази даних. Якщо цільова база даних існує, то вона перезаписується, а якщо ні, то створюється нова
<delete-config>	Видалити конфігураційну базу даних, крім <running/>
<lock>	Заблокувати конфігураційну базу даних на пристрої. Заблокована конфігураційна база даних не може бути змінена іншим менеджером NETCONF. Блокування усуває ризик виникнення помилок, спричинених одночасною зміною конфігурації кількома менеджерами NETCONF, SNMP або CLI-скриптами.
<unlock>	Розблокувати конфігураційну базу даних на пристрої. Менеджер NETCONF не може розблокувати базу даних, яку заблокував інший менеджер NETCONF.
<close-session>	Закрити NETCONF-сесію
<kill-session>	Примусово закрити NETCONF-сесію. Лише адміністратор має права на виконання даної операції
Writable-running	Підтримка прямого запису до бази даних <running>, тобто пристрій підтримує операції <edit-config> та <copy-config> в базі даних <running>
Candidate configuration	Пристрій підтримує конфігураційну базу даних <candidate/>, тобто базу даних для попередньої конфігурації, яку можна змінювати без впливу на поточну конфігурацію. Незалежна база даних <candidate/> створюється для кожної сесії NETCONF

Операція	Опис
Confirmed commit	Дозволяє виконувати операцію <commit>, яка має параметри <confirmed> та <confirm-timeout>. Параметр <confirmed> зберігає всі сконфігуровані дані в базу даних <running> на пристрої. Параметр <confirm-timeout> вказує інтервал часу в секундах, протягом якого діє операція «Confirmed commit» (600 секунд за замовчуванням). Протягом даного проміжку часу діє нова конфігурація і після того, як час вийшов – конфігурація відновлюється до початкової.
Rollback on error	Підтримка можливості «відкату» при виникненні помилок. Якщо виникає помилка і генерується елемент <rpc-error>, то агент NETCONF зупиняє виконання операції <edit-config> і відновлює вказану конфігурацію до стану, який був перед виконанням операції <edit-config>
Distinct startup	Підтримка чіткого запуску пристрою. Агент NETCONF перевіряє доступність і послідовність параметрів перед чітким запуском пристрою. Агент NETCONF підтримує <startup/> конфігураційну базу даних і відрізняє її від бази даних <running/>. Для того щоб зберегти дані конфігурації з бази даних <running/> до бази даних <startup/> необхідно виконати операцію <copy-config>
Notification	Пристрій може відправляти аварії та повідомлення до певної системи управління мережею. Дана операція в протоколі NETCONF базується на з’єднаннях по протоколу TCP
Interleave	Дозволяє покращити ефективність управління шляхом використання однієї NETCONF-сесії для кількох цілей, зменшуючи загальну кількість необхідних NETCONF-сесій

Операція	Опис
Synchronization	Підтримка синхронізації даних з системою управління мережею. NETCONF менеджер може відправляти <rpc> елементи агенту з запитом на синхронізацію даних з пристрою в певну папку призначення. Операція синхронізації підтримує режими <sync-full> та <sync-inc>. Режим <sync-full> синхронізує всі дані з пристрою до папки призанчення, а режим <sync-inc> синхронізує лише вказані дані
Active notification	Пристрій може інформувати свого піра, що він активний. Для операцій, які займають відносно багато часу, менеджер NETCONF може вирішити, що обробка його запиту досягла тайм-ауту і скасувати операцію. Шляхом відправки <active> елементів менеджеру, агент може інформувати його, що запит все ще в процесі обробки і не досяг тайм-ауту
Action	Підтримка операцій по обслуговуванню, таких як очищення лічильника пакетів, перезавантаження плати тощо. Операція <execute-action> запрошує агента запустити певну дію по обслуговуванню. Якщо операція <execute-action> успішна, то агент відправляє елемент <rpc-reply>, який містить в собі значення <ok>. В іншому випадку, агент відправляє елемент <rpc-reply>, який містить в собі значення <rpc-error>

Операція	Опис
Execute CLI	Підтримка виконання CLI-команд на пристрої через протокол NETCONF. Максимум 60 команд можна виконати в рамках одного <rpc> запиту, з максимальним розміром в 512 Байт. Операція <execute-cli> зупиняє виконання команд при виникненні помилки, тобто якщо в процесі виконнаня команд виникне якась помилка, то наступні команди виконуватись не будуть
Update	Означає, що пристрій підтримує оновлення конфігураційних даних. Операція <update> оновлює конфігураційні дані в базі даних <candidate/> з останніми конфігураційними даними з бази даних <running/>, коли виникає помилка під час застосування конфігураційних даних.
Exchange	Означає, що пристрій може взаємодіяти з відправником запиту під час його обробки. Менеджер виконує операцію <get-next>, щоб взаємодіяти з агентом. Наприклад, якщо результат виконання операції <get> або <get-config> включає в себе велику кількість даних, то агент повертає дані використовуючи кілька елементів <rpc-reply>. Після того, як менеджер отримає перший елемент <rpc-reply>, він запрошує в агента наступний елемент <rpc-reply> або скасовує дану послідовність даних.

Протокол NETCONF може використовуватись як в якості «Північого» інтерфейсу, так і в якості «Південного» (Рисунок 2.16). При використанні його в якості «Північого» інтерейсу, необхідно запустити NETCONF на певній системі управління та на контролері для передачі RPC XML-пакетів. Це дає змогу використовувати систему управління для входу та налаштування

контролера. При використанні NETCONF в якості «Південного» інтерфейсу, за допомогою контроллера можна виконувати вхід та налаштування мережевих елементів.

Встановлення з'єднання NETCONF між контролером SDN та мережевими елементами відбувається наступним чином:

- протокол NETCONF активується на контролері SDN і налаштовується наступна інформація про мережевий елемент: IP-адреса, номер порта, ім'я користувача та пароль;
- контролер SDN відправляє мережевому елементу запит RPC на встановлення з'єднання NETCONF. В пакеті RPC-запиту міститься інформація про IP-адресу мережевого елемента;
- після отримання запиту RPC, мережевий елемент перевіряє, чи співпадає IP-адреса, що міститься в пакеті, з його власною IP-адресою. Якщо вони співпадають, то мережевий елемент відповідає контроллеру повідомленням RPC-reply для встановлення зв'язку NETCONF.

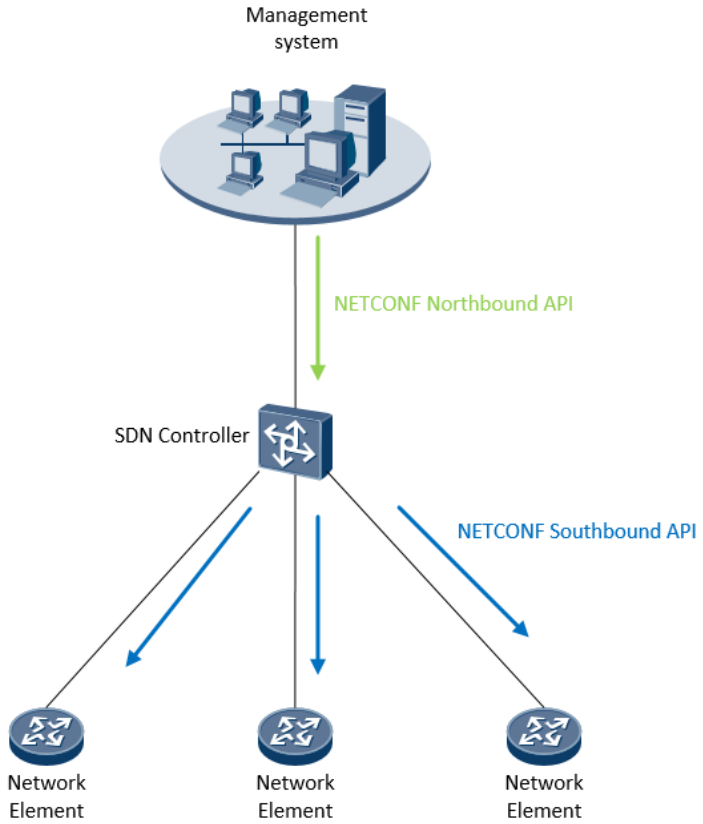

Рисунок 2.16. Застосування протоколу NETCONF.

2.3.Порівняння ефективності роботи протоколів NETCONF та OpenFlow

Базуючись на описі принципів роботи протоколів OpenFlow та NETCONF, практично неможливо порівняти їх за певними критеріями чи показниками, оскільки логіка роботи та архітектура даних протоколів різна. Необхідно базуватись на результатах тестувань роботи даних протоколів для конкретних випадків застосування техноголої SDN, наприклад SD-WAN, CloudVPN, SD-DCN тощо.

Т. Кунц та К. Мутукумар з Карлтонського Університету провели дослідження, в якому порівняли ефективність роботи протоколів NETCONF та OpenFlow за такими показниками: тривалість виконання операції, ефективність використання наявної ємності каналів та кількість повідомлень необхідних для виконання певної операції.

Дослідження проводились на мережі, яка складається з двох пристроїв оптичної транспортної мережі Juniper BTI7800, які з’єднують між собою два віддалених цетри обробки даних (Рисунок 2.17). Елементи BTI7800 з’єднані і контролюються SDN-контроллером OpenDayLight, який реалізує функцію BoD, тобто забезпечення необхідної величини смуги пропускання у конкретний момент часу. Контролер SDN розгорнуто на сервері, який працює на операційній системі CentOS. На цьому ж сервері запущено додаток BoD, який написаний на мові Python. Менеджер ресурсів збирає інформацію про пристрої та їх топологію, використовуючи «Північні» інтерфейси REST. Додаток BoD отримує запит на виділення пропускної здатності від користувача і передає цю інформацію менеджеру ресурсів. Менеджер ресурсів на основі збереженої інформації та залежно від характеру трафіку обробляє сценарії BoD, переводячи вимоги користувача як REST API інформацію до площини управління SDN. REST API містить інформацію, яка однозначно ідентифікує підключені пристрої та зміни їх з'єднань. Елементи BTI7800 з’єднані між собою оптичним каналом з ємністю 100 Гбіт/с, а в сторону центру обробки даних кожен елемент BTI7800 має по дванадцять портів ємністю 10

Гбіт/с кожен, тобто обслуговує 12 користувачів і кожен з них має смугу шириною 10 Гбіт/с. Елемент Spirent виступає в якості генератору трафіку. Генератором трафіку Spirent не можна керувати за допомогою контролера SDN, але він завжди генеруює трафік на основі вказаних конфігурацій. Прикладний рівень контролера пропонує інтерактивний режим вибору сценарію використання, а також сигналізує площину управління про тип трафіку та про рішення зміни потоку. Елемент BTI7800 отримує на вхід 120 Гбіт/с трафіку зі сторони одного центру обробки даних, але може передати на інший елемент BTI7800 лише 100 Гбіт/с. SDN-контроллер повинен вирішити задачі виділення необхідної ширини смуги пропускання для користувачів відповідно до вказаних вимог.

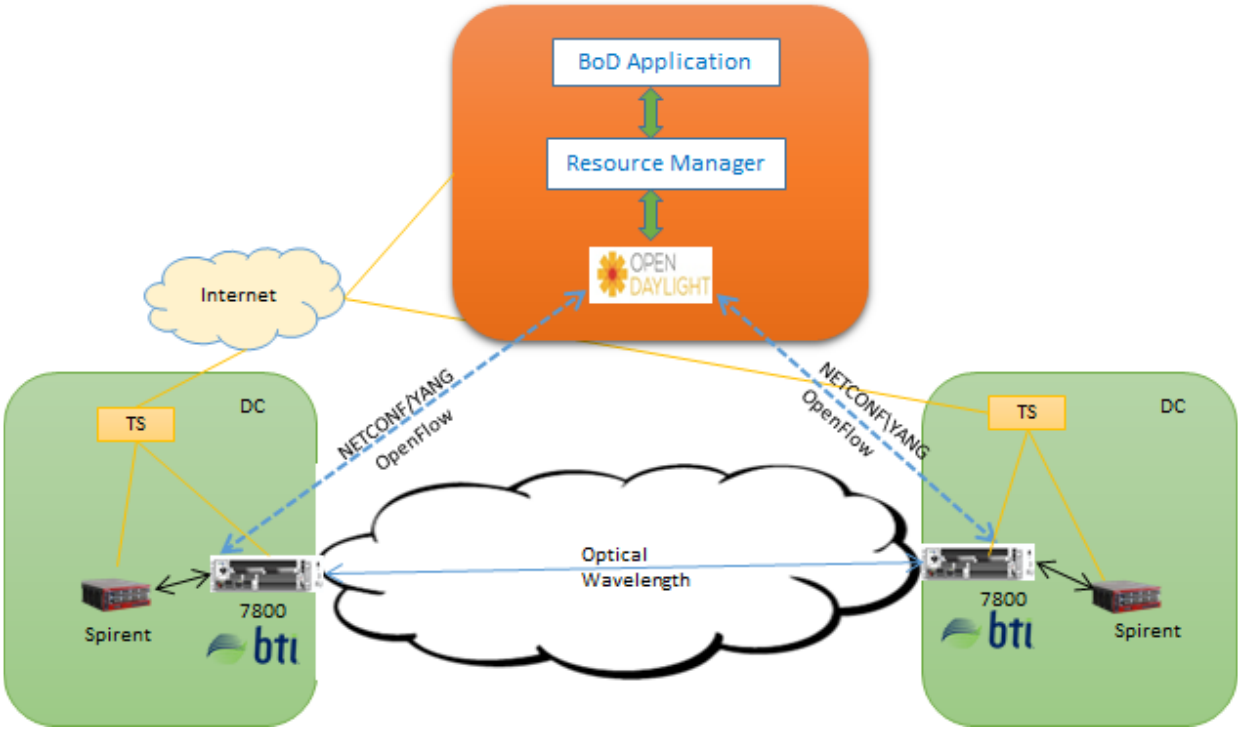

Рисунок 2.17. Схема мережі для тестування протоколів NETCONF та OpenFlow.

Для порівняння ефективності протоколів було проаналізовано кілька сценаріїв їх використання. Дані сценарії були підібрані для функції BoD і включають в себе:

- потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача Б;

- потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача А;
  - коли немає потоку низького пріоритету;
  - коли є потік низького пріоритету;
- потік високого пріоритету для обох користувачів зі взаємним поділом;
- запит на смугу шириною 40 Гбіт/с;
- терміновий запит на смугу шириною 100 Гбіт/с;
- стрес-тест, тобто коли кілька додатків намагаються змінити конфігураційні дані на одному й тому ж пристрої в однаковий час.

Для оцінки ефективності були використані наступні показники:

- час: ефективний протокол повинен забезпечувати швидку і надійну роботу. Порівнювався проміжок часу, який необхідний для виконання вказаного набору конфігурацій обома протоколами. Період часу для кожного випадку використання відрізняється залежно від того, коли система досягає сталого стану, тобто коли всі конфігураційні команди зберігаються в поточному файлі конфігурації елементу і застосовуються, а потоки трафіку успішно функціонують за зміненими правилами;
- використання пропускної здатності: для конкретного сценарію, використання пропускної здатності обчислюється протягом періоду часу, відколи користувач починає використовувати даний сценарій і поки система не перейде до стаціонарного стану. Усі досліди починались з приведення роботи системи до визначеного початкового стану. Новий користувач вимагає додаткової пропускної здатності і контролер SDN вирішує обслуговувати запити на виділення пропускної здатності, базуючись на типу трафіку та наявності ресурсів. Площина управління SDN повідомляє про зміни через обраний «Південний» протокол і чекає, поки система не виконає вимоги. Після виконання вимог система повертається до початкового стану. У стаціонарному стані, якщо всі канали є активними, використання пропускної здатності завжди становить 100%.

- повідомлення управління: контролер SDN управляє пристроями передачі даних через реалізований «Південний» протокол, який передає інформацію про зміни потоків за допомогою контрольних повідомлень. Кожен «Південний» протокол обробляє повідомлення по-різному, що призводить до різниці у розмір повідомлень та їх кількості. Ефективний протокол повинен вимагати менше керуючої сигналізації і, отже, використовувати менше мережевих ресурсів.

З описаних вище параметрів, час і пропускна здатність однаково важливі для досягнення QoS серед користувачів та різного типу трафіку. Повідомлення управління не мають такого великого значення, але використання ресурсів мережі, які необхідні для реалізації сигналізації, потрібно враховувати при проектуванні системи.

На Рисунку 2.18 зображена топологія, яка використовується для тестування всіх сценаріїв.

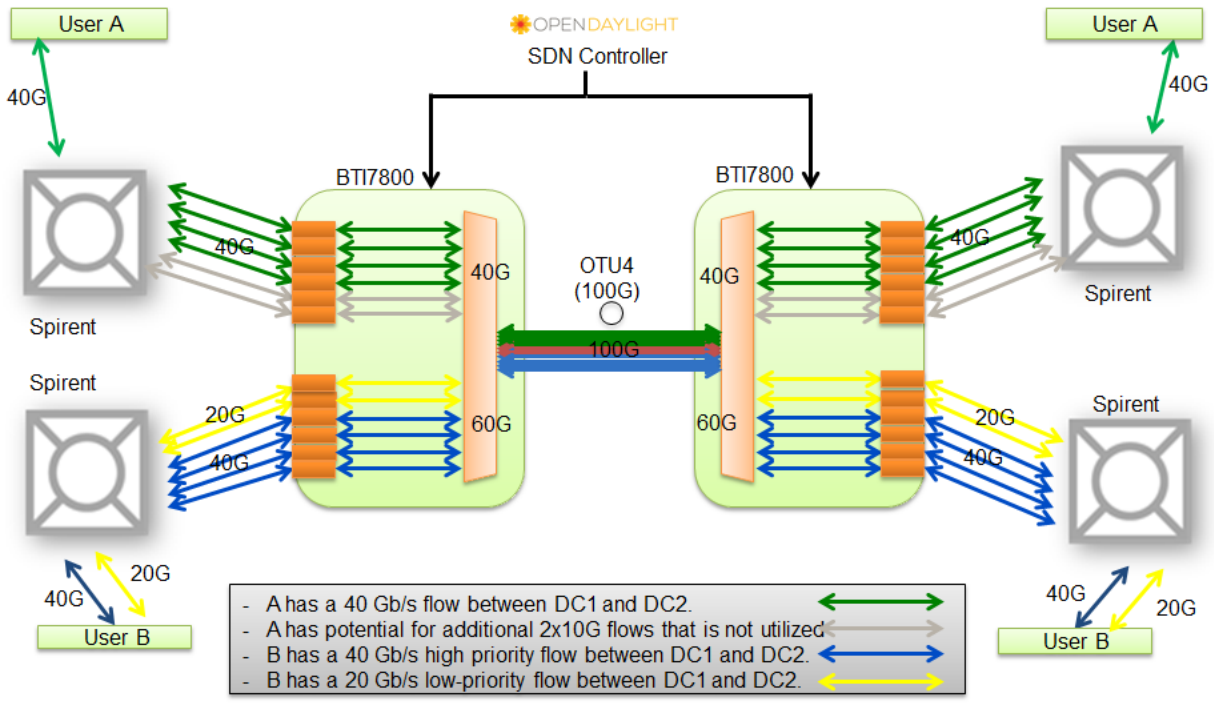

Рисунок 2.18. Топологія для тестування обраних сценаріїв.

Обидва центри обробки даних мають однакову топологію внутрішньої мережі. В обох центрах обробки даних пристрій BTI7800 виконує роль крайового вузла. Кожен центр даних має двох користувачів, які підключені до BTI7800.

В кожному центрі обробки даних є по два користувача: користувач А та користувач Б. Кожен користувач має потенційну пропускну здатність ємністю 60Гбіт/с, а саме шість каналів ємністю по 10Гбіт/с кожен, які з'єднані з BTI7800. Кожен користувач має виділену пропускну здатність ємністю 40Гбіт/с, що становить 80 Гбіт/с із 100Гбіт/с доступної пропускної здатності (сині та зелені стрілки). Один з користувачів, «користувач Б», запросив решту вільної ємності величиною 20Гбіт/с для трафіку з низьким пріоритетом (стрілка жовтого кольору). Стрілки, зображені на BTI7800, відомі як перехресні з'єднання або підпотоки. Вони реалізують мультиплексування будь-яких 10 із 12 портів для передачі на вихід OTU4. Базуючись на тому, який саме вид трафіку використовується користувачем, контролер SDN сигналізує які 10 портів необхідно перехресно скомутувати до OTU4.

На Рисунку 2.19 зображено час, необхідний для встановлення початкового середовища для схеми, яка зображена на Рисунку 2.18, при використанні в якості «Південнних» протоколів NETCONF та OpenFlow.

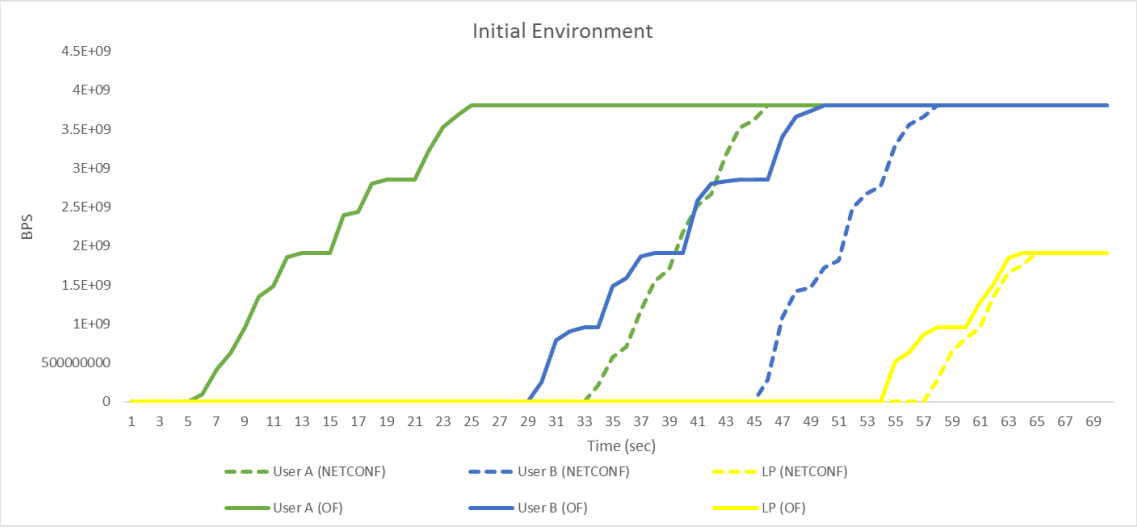

Рисунок 2.19 Встановлення початкового середовища при використанні протоколів NETCONF ТА OpenFlow.

Для реалізації передачі потоків трафіку між центрами обробки даних важливо забезпечити тунель «від кінця-до кінця». Прикладний рівень спочатку опрацьовує центр обробки даних на одному кінці тунелю, але на іншому кінці ще не сконфігуровано з'єднання. Тому трафік не може досягти іншого кінця. Далі прикладний рівень починає опрацьовувати центр обробки даних на іншому кінці. В результаті утворюється тунель «від кінця-до кінця» для кожного підпотоку, і в результаті трафік починає передаватись по даному тунелю.

Модель YANG обробляє всі перехресні з'єднання як єдиний набір даних. Контролер не дозволяє отримати доступ до них як до окремих перехресних з'єднань і змушує отримувати доступ до них як до набору перехресних з'єднань. Протокол NETCONF вказує список перехресних з'єднань, які потрібно реалізувати, а пристрій BTI7800 отримує запит і ітеративно обробляє інформацію про перехресні з'єднання, вказані у списку. В кінці, він повертає статус операції назад контролеру. На Рисунку 2.20 зображені всі кроки, які необхідні для встановлення початкового середовища при використанні протоколу NETCONF.

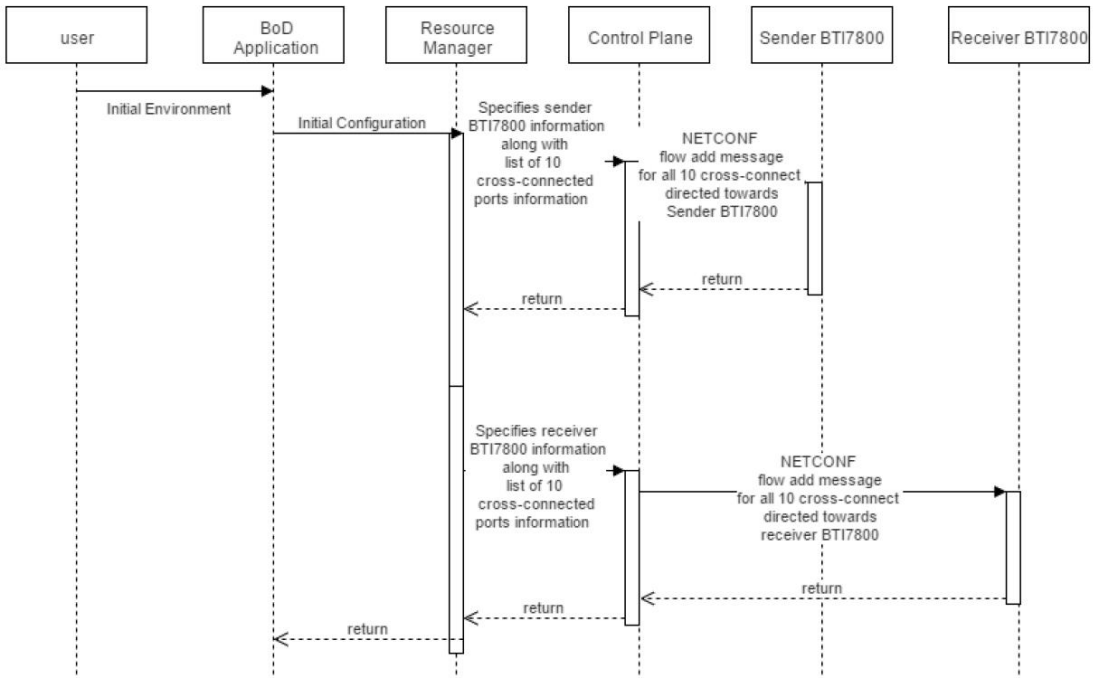

Рисунок 2.20. Етапи встановлення початкового середовища при використанні протоколу NETCONF.

1. Користувач за допомогою додатка BoD запрошує контролер SDN встановити початкове середовище.

2. Менеджер ресурсів отримує запит користувача від додатку BoD і визначає перелік підпотоків або перехресних з'єднань, необхідних для організації на обох пристроях BTI7800.

3. Менеджер ресурсів надає контрольній площині інформацію про пристрій відправника BTI7800 та список усіх десяти перехресних з'єднань, які необхідно реалізувати.

4. Площина управління переводить отриману інформацію у вигляді повідомлення NETCONF <edit-config>, яке відправляється агенту NETCONF відправника BTI7800.

5. Агент NETCONF, що знаходиться на пристрої BTI7800, обробляє запитувані перехресні з'єднання. Після налаштування запитуваних перехресних з'єднань він відправляє повідомлення про успішне завершення, яке передається менеджеру ресурсів.

6. Менеджер ресурсів отримує запит користувача від додатку BoD і визначає перелік підпотоків або перехресних з'єднань, необхідних для організації на обох пристроях BTI7800.

7. Кроки 4 і 5 повторюються знову для приймача BTI7800.

У момент часу t = 1 сек на графіку, представленому на Рисунку 2.19, контролер SDN передає набір перехресних з’єднань, які мають бути реалізовані на пристрої відправника BTI7800. На початку перехресні з’єднання ще не створені, але в момент часу t = 31 сек перехресні з'єднання на пристрої відправника BTI7800 вже реалізовані. Контролер, використовуючи реалізований протокол NETCONF, передає інформацію про перехресні з'єднання, які необхідно реалізувати, пристрою BTI7800 на приймальному кінці. Трафік успішно приймається користувачем кожного каналу з моменту часу t = 32 сек до t = 70 сек. В момент часу t = 34 сек встановлюється канал

для першого підпотоку. В момент часу t = 44 сек встановлюється канал для базового потоку користувача А. На момент часу t = 63 сек всі канали є успішно встановленими. Протягом часу з t = 63 сек до t = 70 сек система перебуває в стаціонарному стані, в якому всі 10 каналів є активними і повністю утилізованим.

Використання пропускної здатності обчислюється від початку експерименту при t = 1 сек до t = 70 сек, поки система не набуде стаціонарного стану і перебуватиме в ньому протягом декількох секунд. Використання пропускної здатності протягом даного проміжку часу становить 32,01%.

Протокол OpenFlow, на відміну від NETCONF, працює окремо з кожним підтоком. Оскільки кожен потік обробляється окремо, то величина часу, коли канал не використовується, є меншою у протоколу OpenFlow, порівняно з NETCONF. На Рисунку 2.21 зображені всі кроки, які необхідні для встановлення початкового середовища при використанні протоколу OpenFlow.

1. Користувач за допомогою додатка BoD запрошує контролер SDN встановити початкове середовище.

2. Менеджер ресурсів отримує запит користувача від додатку BoD і визначає перелік підпотоків або перехресних з'єднань, необхідних для організації на обох пристроях BTI7800.

3. Менеджер ресурсів надає контрольній площині інформацію про пристрій відправника BTI7800 та список усіх десяти перехресних з'єднань, які необхідно реалізувати.

4. Площина управління переводить отриману інформацію як повідомлення OpenFlow-Add, і передає його відправнику пристроя BTI7800.

5. Пристрій відправника BTI7800, отримуючи повідомлення OpenFlow-Add, забезпечує нове перехресне з'єднання та надсилає повідомлення про завершення.

6. Менеджер ресурсів надає контрольній площині інформацію про приймач BTI7800 і перший підпотік або перехресне з'єднання, які необхідно реалізувати.

7. Площина управління переводить отриману інформацію у вигляді повідомлення OpenFlow-Add і надсилає його приймачу BTI7800.

8. Приймач пристрою BTI7800, отримуючи повідомлення OpenFlow-Add, забезпечує нове перехресне з'єднання та надсилає повідомлення про завершення.

9. Кроки з третього по восьмий повторюються для решти дев’яти підпотоків або перехресних з’єднань.

В момент часу t = 1 сек на графіку, представленому на Рисунку 2.19, контролер SDN передає інформацію першого підпотоку в центр обробки даних відправника і така ж інформація передається в інший центр обробки даних. В момент часу t = 6 сек канал першого підпотоку встановлюється і трафік відправляється з одного центру обробки даних і успішно приймається іншим центром обробки даних. В момент часу t = 61 сек усі десять підпотоків встановлюються між обома центрами обробки даних і трафік у всіх каналах активний. Протягом часу від t = 61 сек до t = 70 сек система перебуває в стаціонарному стані з усіма активними і повністю утилізованими десятьма каналами.

Протокол OpenFlow демонструє використання пропускної здатності на 53,89%, що більше, ніж результат протоколу NETCONF.

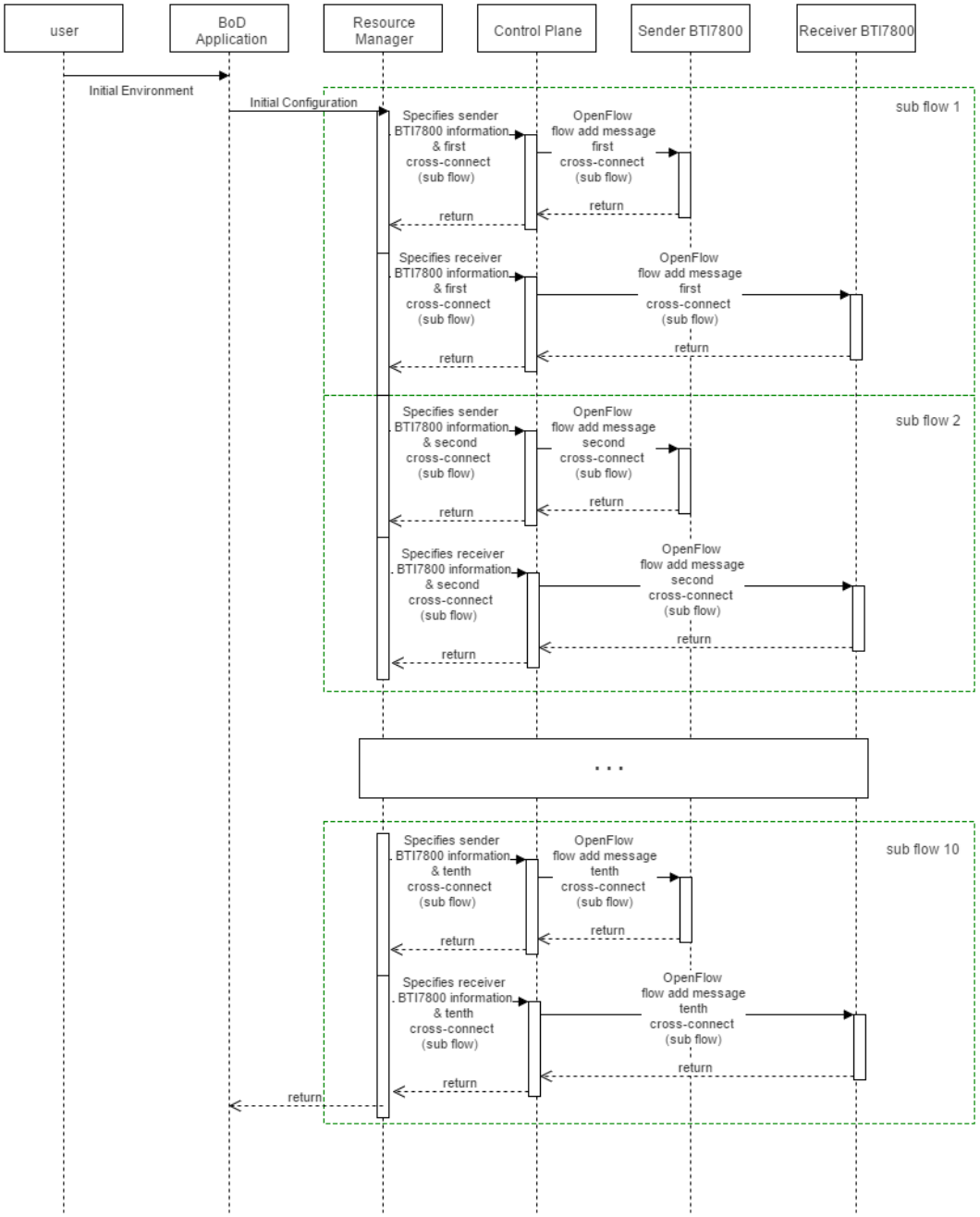

Рисунок 2.21. Етапи встановлення початкового середовища при використанні протоколу OpenFlow.

Для порівняння обох протоколів було обрано набір сценаріїв, в яких обслуговуються по два користувача в кожному центрі обробки даних під управлінням одного контролера SDN. Дані сценарії орієнтовані на обробку запитів на виділення пропускної здатності для трафіку з високим пріоритетом. У всіх розглянутих сценаріях користувач запрошуватиме смугу шириною 20Гбіт/с для передачі трафіку з високим пріоритетом. Залежно від випадку використання він може змінюватися, якщо лише один з користувачів або

обидва користувачі просять пропускну здатність для потоків з високим пріоритетом.[4]

2.3.1.Потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача Б

Користувачу Б потрібно 20Гбіт/с пропускної здатності для трафіку з високим пріоритетом і користувач Б не має трафіку з низьким пріоритетом. Смуга шириною 20Гбіт/с, яка була призначених для трафіку користувача Б з низьким пріоритетом, тепер використовується для передачі трафіку з високим пріоритетом, як показано на Рисунку 2.22.

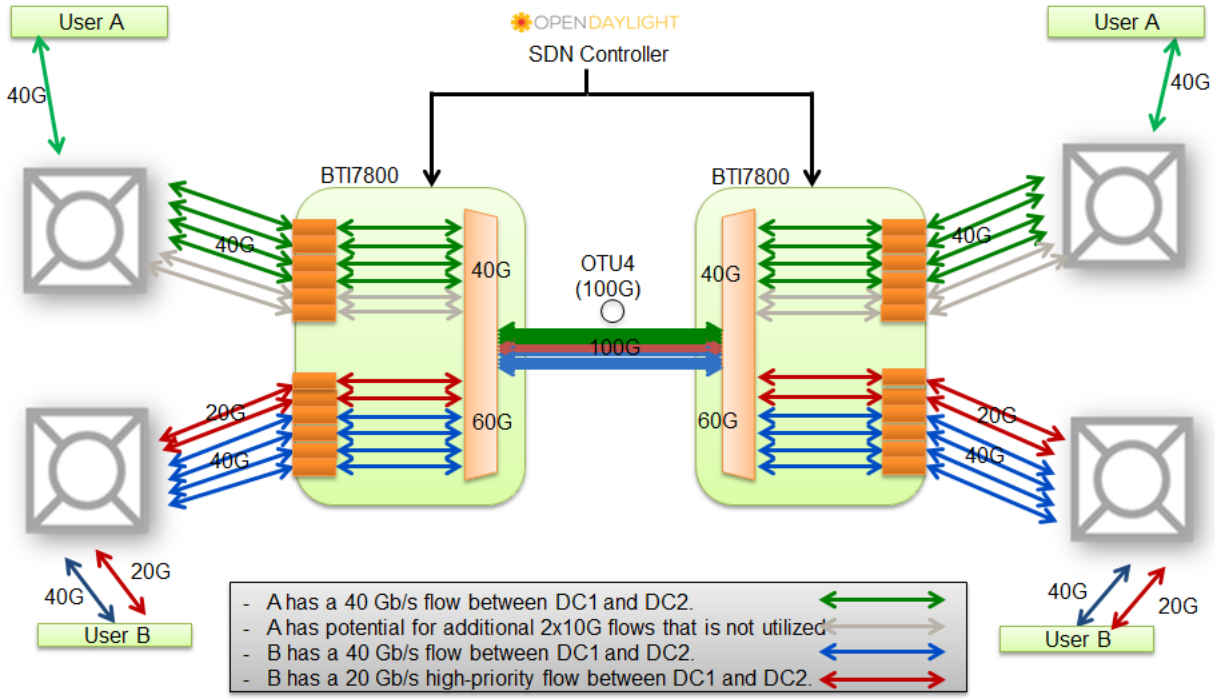

Рисунок 2.22.Потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача Б.

Ресурси для трафіку з низьким пріоритетом призначаються користувачу Б і тому ж користувачеві потрібна пропускна здатність для трафіку з високим пріоритетом. Тому трафік з високим пріоритетом використовуватиме пропускну здатність, призначену для трафіку з низьким пріоритетом. Користувач взаємодіє з додатком BoD і запит на виділення пропускної

здатності для високопріоритетного трафіку користувача Б передається менеджеру ресурсів, як показано на Рисунку 2.23. Менеджер ресурсів не ініціює змін в даному випадку.

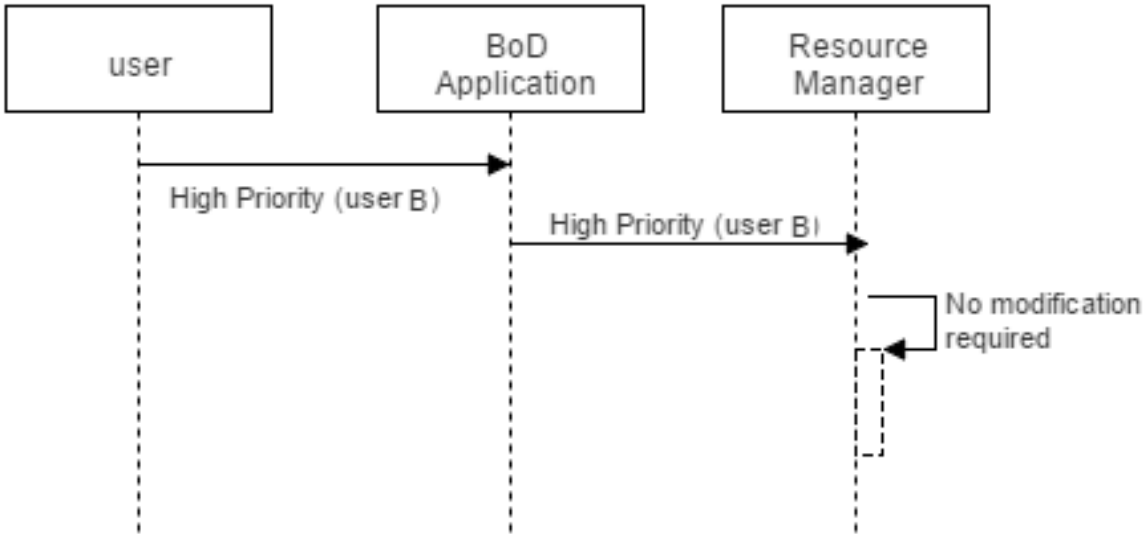

Рисунок 2.23. Етапи надсилання запиту на виділення смуги для високопріоритетного трафіку користувача Б.

В даному сценарії сумарна пропускна здатність становить 100Гбіт/с і всі вимоги щодо пропускної здатності виконані (Таблиця 2.3).[4]

Таблиця 2.3.Результати обслуговування користувачів.

Користувач	**Ширина смуги**	**Тип потоку**	**Результат**
Користувач А	40Гбіт/с(Зелений)	Базовий потік	Забезпечено
Користувач Б	40Гбіт/с(Синій)	Базовий потік	Забезпечено
	20Гбіт/с(Червоний)	Високий пріоритет	Забезпечено

2.3.2.Потік високого пріоритету з ємністю каналу 20Гбіт/с для користувача А

Обом користувачам виділено по 40Гбіт/с смуги пропускання для базових потоків. Залишається вільна смуга шириною 20Гбіт/с на каналі, що з'єднує центри обробки даних між собою. Користувач А надсилає запит на

виділення смуги пропускання ємністю 20Гбіт/с для високопріоритетного трафіку. Контролер виділяє користувачеві А смугу шириною 20Гбіт/с, як показано на Рисунку 2.24.

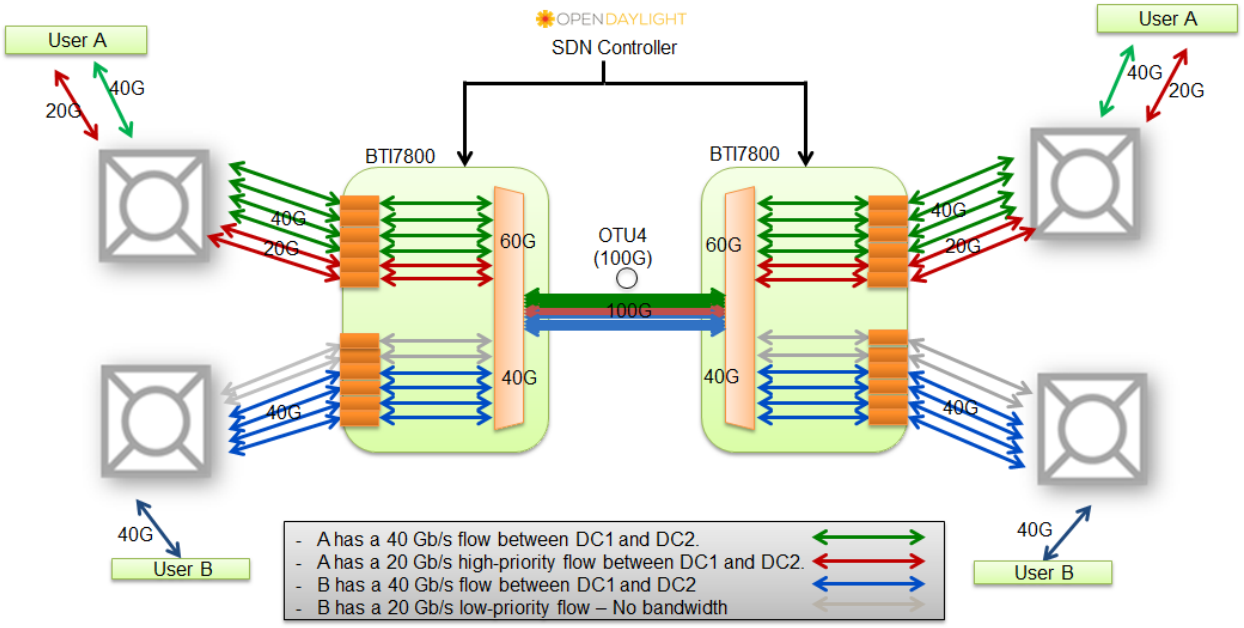

Рисунок 2.24. Потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача А – перший випадок.

На Рисунку 2.25 зображена послідовність кроків, що виконуються при налаштуванні запиту високого пріоритету за допомогою протоколу NETCONF. Користувач взаємодіє з додатком BoD, а менеджер ресурсів отримує запит. Система перебуває в початковому стані з 8 реалізованими підпотоками чи перехресними з'єднаннями.

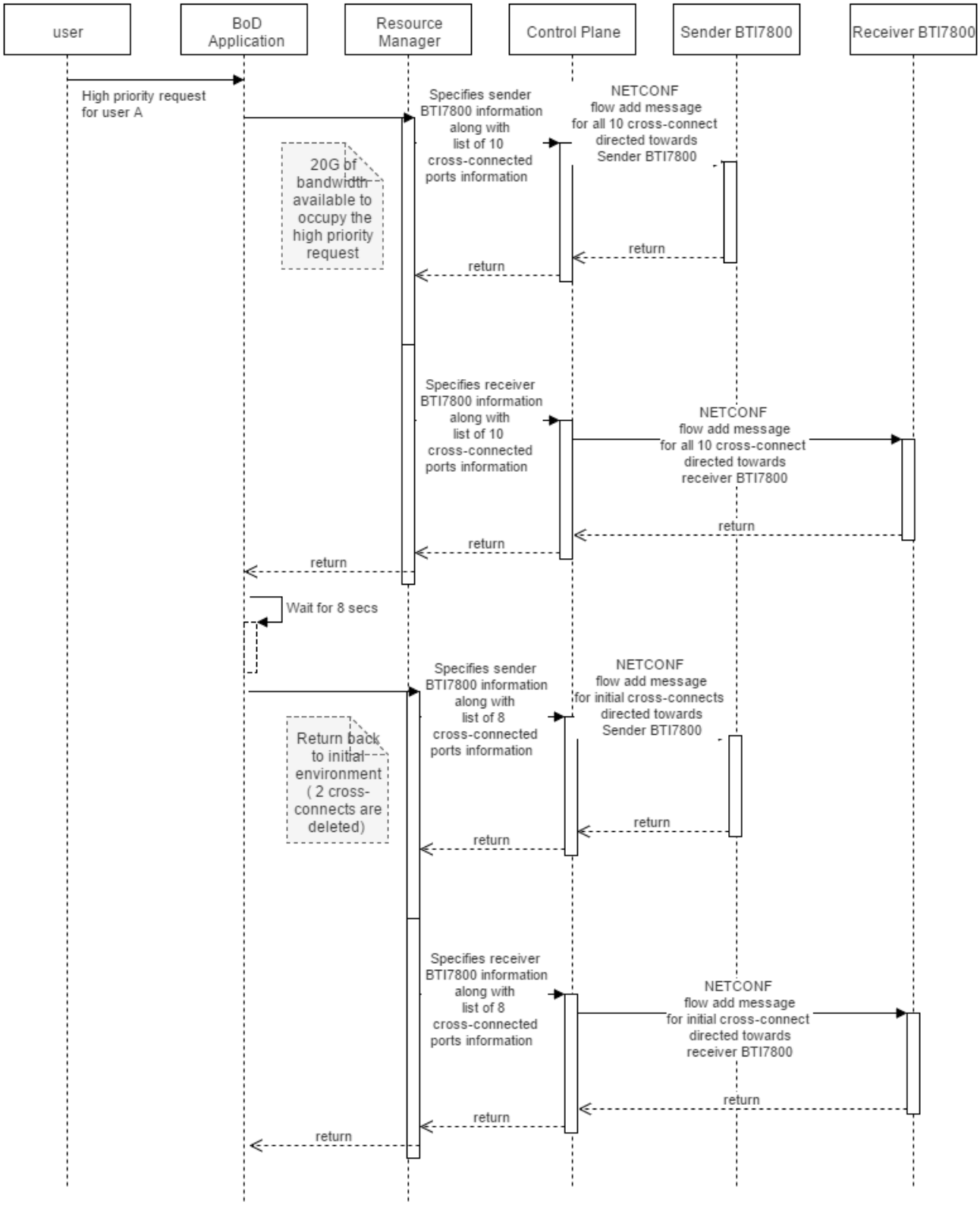

Рисунок 2.25. Послідовність кроків, що виконуються при налаштуванні запиту високого пріоритету за допомогою протоколу NETCONF – перший випадок.

Користувач А надіслав запит на 2 додаткових підпотоки для обробки трафіку з високим пріоритетом. Менеджер ресурсів передає набір з 10 перехресних з'єднань (існуючих 8 перехресних з'єднань, що належать до

початкового середовища, та додаткових двох перехресних з'єднань.) на передавач пристрою BTI7800. Після завершення передавач пристрою BTI7800 поширює інформацію про успішність. Менеджер ресурсів передає ту саму інформацію до приймача BTI7800. Додатку BoD відомо, що перехресні з'єднання реалізовані для задоволення вимог високого пріоритету. Через 8 секунд додаток BoD подає сигнал менеджеру ресурсів повернутися до початкової конфігурації. Менеджер ресурсів надсилає набір з 8 перехресних з'єднань або підпотоків, що належать до початкової конфігурації, на обидва пристрої BTI7800. В результаті, перехресні з'єднання, створені для обробки високопріоритетного трафіку користувача A, руйнуються.

На Рисунку 2.26 зображена послідовність кроків, що виконуються при налаштуванні запиту високого пріоритету за допомогою протоколу OpenFlow. Користувач взаємодіє з додатком BoD, а менеджер ресурсів отримує запит. Система перебуває в початковому стані з 8 реалізованими підпотоками чи перехресними з'єднаннями. Користувач A надіслав запит на 2 додаткових підпотоки для обробки трафіку з високим пріоритетом. Менеджер ресурсів передає інформацію 9-го перехресного з’єднання або підпотоку передавачу та приймачу пристроїв BTI7800. Після того, як 9-й підпотік успішно створено на обох пристроях BTI7800, - передається інформація про 10-й підпотік. Додаток BoD знає, що підпотоки встановлюються для задоволення пропускної здатності з високим пріоритетом. Через 8 секунд додаток BoD подає сигнал менеджеру ресурсів про повернення до початкової конфігурації. Менеджер ресурсів через площину управління сигналізує, що 9-те перехресне з'єднання або підпотік потрібно видалити на обох пристроях BTI7800. Те ж саме відбувається для видалення 10-го підпотоку.

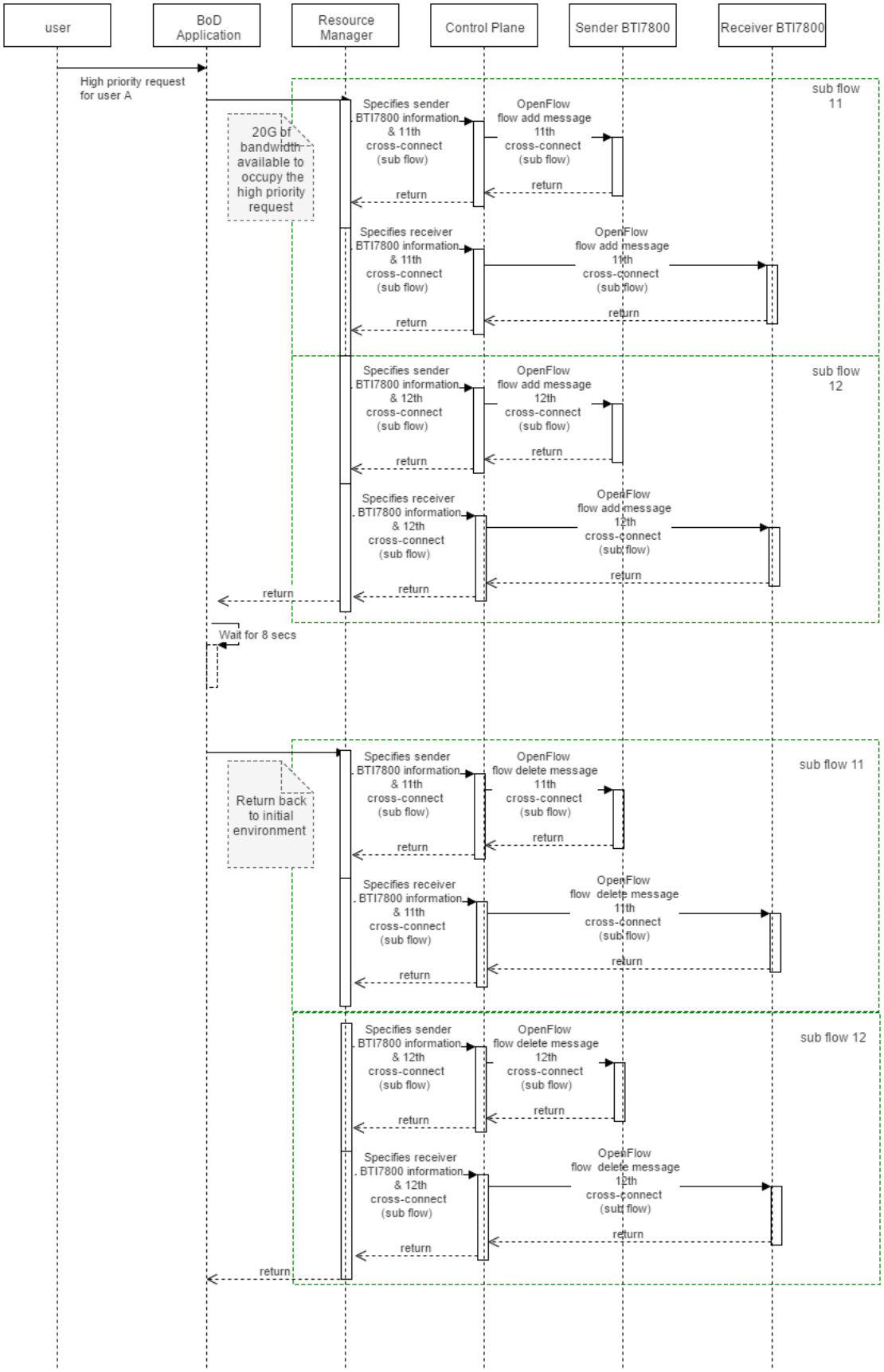

Рисунок 2.26. Послідовність кроків, що виконуються при налаштуванні запиту високого пріоритету за допомогою протоколу OpenFlow.

Результати роботи протоколів NETCONF та OpenFlow представлені на Рисунку 2.27 та Рисунку 2.28 відповідно. Дослідження починається починається в момент часу t = 10 сек, а система знаходиться в початковому стані. При t = 10 сек прикладний рівень сигналізує менеджеру ресурсів про запит високого пріоритету. При роботі протоколу NETCONF, налаштування 9-го підпотоку закінчується в момент часу t = 18 сек , а налаштування 10-го підпотоку в момент часу t = 22 сек. З моменту часу t = 22 сек і до t = 30 сек вимоги щодо смуги для високопріоритетного трафіку задовольняються.

При роботі протоколу OpenFlow, налаштування 9-го підпотоку закінчується в момент часу t = 17 сек, а налаштування 10-го підпотоку в момент часу t = 22 сек. З моменту часу t = 22 сек і до t = 30 сек вимоги щодо смуги для високопріоритетного трафіку задовольняються.

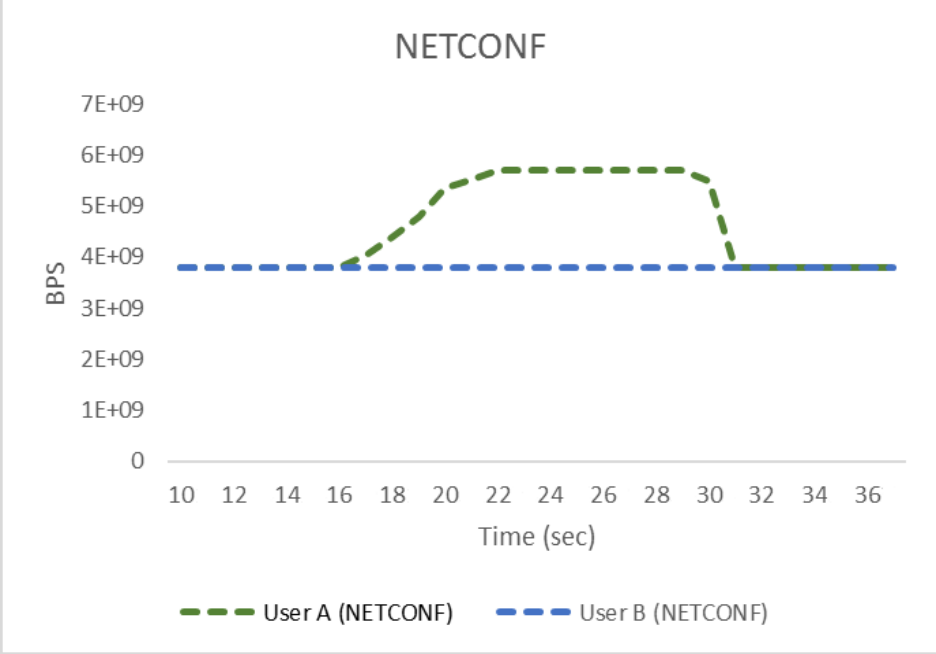

Рисунок 2.27. Реалізація запиту на смугу пропускання для високопріоритетного трафіку користувача А, використовуючи протокол NETCONF – перший випадок.

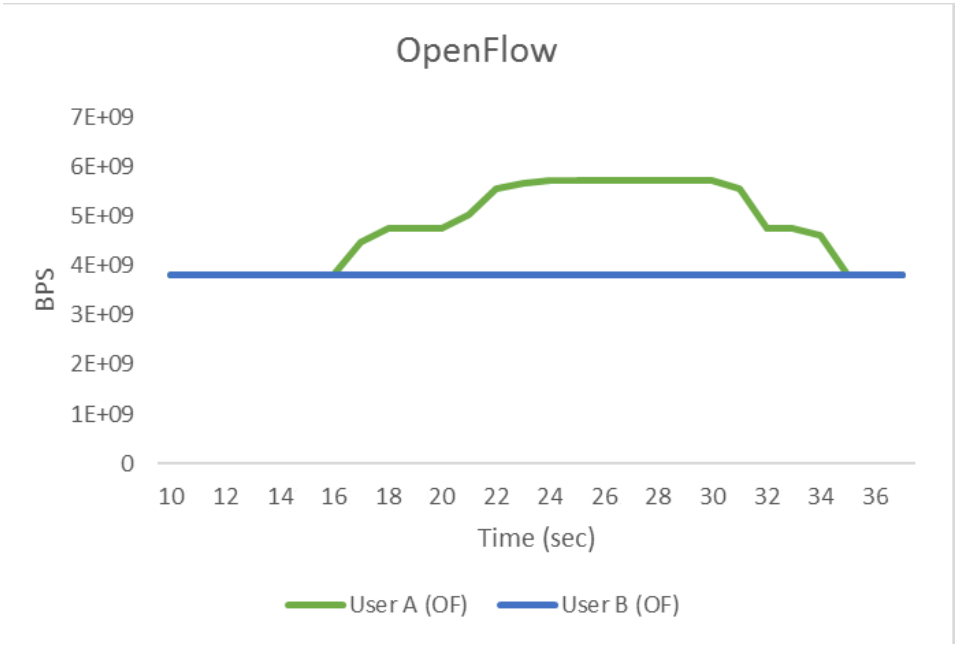

Рисунок 2.28. Реалізація запиту на смугу пропускання для високопріоритетного трафіку користувача A, використовуючи протокол OpenFlow – перший випадок.

Для протоколів NETCONF і OpenFlow потрібно 12 секунд, щоб обробити два перехресних з'єднання на кожному кінці каналу, 3 секунди щоб налаштувати обладнання для кожного перехресного з'єднання. В момент часу t = 31 сек при використанні протоколу NETCONF, починається процес повернення до початкового стану середовища і для видалення обох перехресних з'єднань потрібно 0,5 сек. В момент часу t = 31 сек при використанні протоколу OpenFlow починається процес повернення до початкового стану середовища і для видалення кожного перехресного з'єднання потрібно 0,75 сек. У випадку OpenFlow для кожного перехресного з'єднання, яке потрібно видалити, необхідно надіслати окреме повідомлення, що призводить до збільшення використання мережевих та апаратних ресурсів.

Використання пропускної здатності для цього сценарію обчислюється для моменту часу, коли потік високого пріоритету починає обслуговуватися і поки він не закінчиться. Обидва протоколи повинні обслуговувати запит з високим пріоритетом протягом 8 секунд, але вікно для трафіку з високим пріоритетом включає в себе налаштування підпотоків. В обох протоколах

високопріоритетні потоки починаються в момент часу t = 16 сек. Протокол NETCONF має вікно тривалістю 16 секунд, а використання пропускної здатності становить 94,4%. Протокол OpenFlow має вікно тривалістю 20 секунд, а використання пропускної здатності становить 93,37%. Хоча різниця дуже мала, проте протокол NETCONF має кращий результат, ніж протокол OpenFlow.

Розглянемо другий випадок, в якому користувач А має високопріоритетний трафік і потребує додаткової пропускної здатності. Користувач Б, який має ресурси для низькопріоритетного трафіку, втрачає виділену йому пропускну здатність, а користувачеві А присвоюється дана пропускна здатність, щоб задовольнити високі вимоги щодо високопріоритетного трафіку, як показано на Рисунку 2.29.

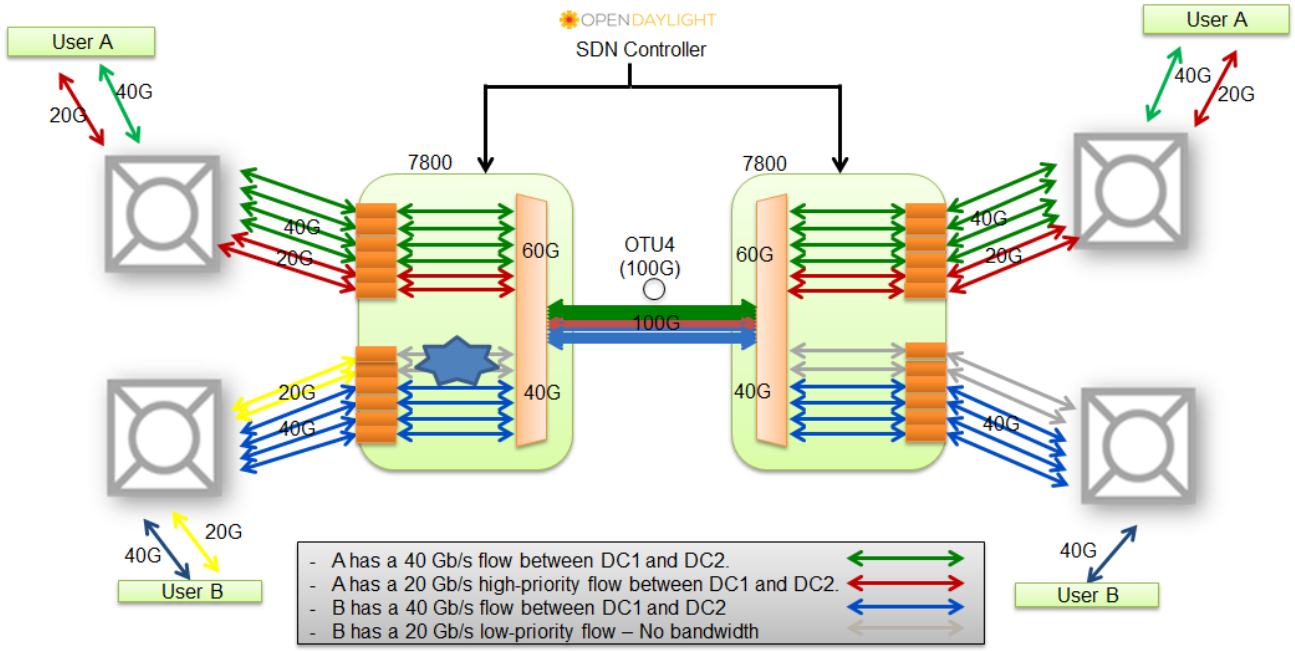

Рисунок 2.29. Потік високого пріоритету з ємністю каналу 20 Гбіт/с для користувача А – другий випадок.

Після виконання вимог щодо високопріоритетного трафіку середовище повертається до початкового стану. Контролер звільнить пропускну здатність, зайняту високопріоритетним трафіком користувача А, і виділить її назад для низькопріоритетного трафіку користувача Б.

Користувачі А і Б мають виділену пропускну здатність по 40Гбіт/с кожен, а 20Гбіт/с трафіку з низьким пріоритетом призначається користувачу Б. Користувач А має трафік з високим пріоритетом, і він потребує розширення поточної смуги пропускання. Контролер SDN подасть сигнал мережевому елементу BTI7800 на зменшення пропускної здатності, яка була виділена трафіку з низьким пріоритетом, і виділить пропускну здатність для трафіку з високим пріоритетом користувачеві А на періоду часу 8 секунд. Інформація про модифікацію потоку передається прикладним рівнем до площини управління. В даному сценарії з’являється запит на ширину смуги 120Гбіт/с, при доступних 100Гбіт/с. В результаті контролер приймає рішення про виділення смуги пропускання для потоків, виходячи з характеру трафіку, як це показано в Таблиці 2.4.

Таблиця 2.4. Виділення смуги пропускання для трафіку користувача А з високим пріоритетом.

Користувач	**Ширина смуги**	**Тип потоку**	**Результат**
Користувач А	40Гбіт/с	Базовий потік	Забезпечено
	20Гбіт/с	Високий пріоритет	Забезпечено
Користувач Б	40Гбіт/с	Базовий потік	Забезпечено
	20Гбіт/с	Низький пріоритет	Недостатньо ресурсів

На Рисунку 2.30 та Рисунку 2.31 представлено результати обробки потоків для протоколів NETCONF та OpenFlow. Дослідження починаються в момент часу t = 10 сек., середовище перебуває в початковому стані. У момент часу t = 10 сек прикладний рівень надсилає менеджеру ресурсів запит на виділення пропускної здатності для високопріоритетного трафіку користувача А. Для цього необхідно видалити та додати два перехресних з'єднання. Для протоколів NETCONF і OpenFlow дані операції закінчуються в моменти часу t = 23 сек і t = 25 сек, відповідно. Для реалізації запиту на забезпечення смуги для високопріоритетного трафіку протоколи NETCONF та OpenFlow

потребують 13 сек і 15 сек, відповідно. Протокол NETCONF потребує 6,5 сек для обслуговування одного центру обробки даних, 0,5 сек для видалення обох перехресних з'єднань та 3 секунди для додавання крос-з'єднання. Протокол OpenFlow потребує 3 сек, щоб додати перехресне з'єднання і 0,75 сек для видалення одного перехресного з'єднання. Протокол OpenFlow потребує більше часу в порівнянні з NETCONF, оскільки зміна кожного підпотоку виконується окремо. Менеджер ресурсів повідомляє, який конкрето підпотік потрібно змінювати один за одним.

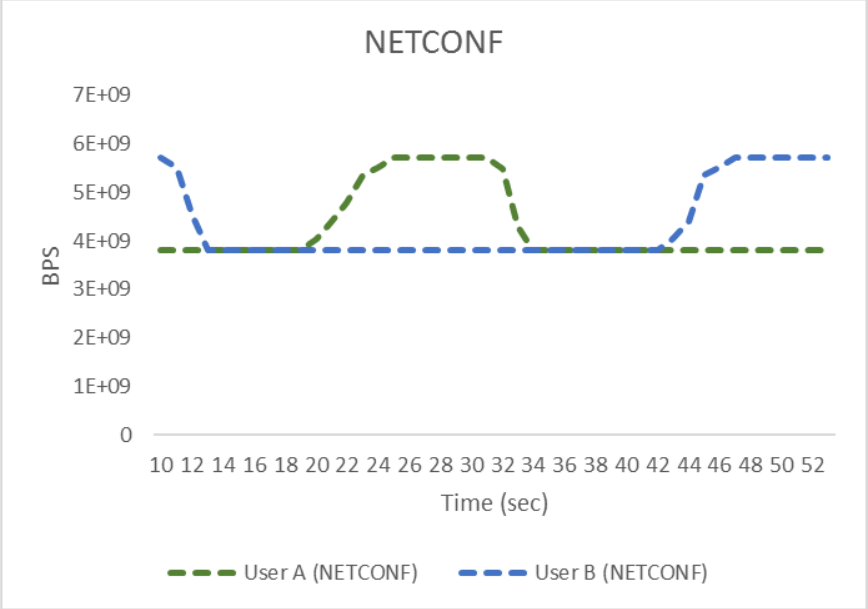

Рисунок 2.30. Реалізація запиту на смугу пропускання для високопріоритетного трафіку користувача А, використовуючи протокол NETCONF – другий випадок.

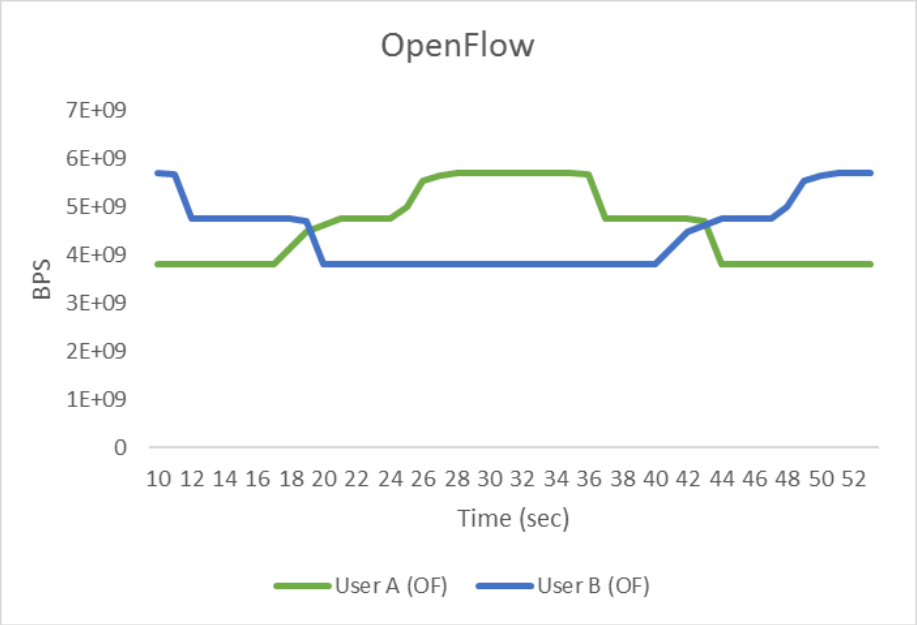

Рисунок 2.31. Реалізація запиту на смугу пропускання для високопріоритетного трафіку користувача А, використовуючи протокол OpenFlow– другий випадок.

Протокол NETCONF використовує пропускну здатність на 92,78%, протокол OpenFlow - на 95,85%. Протокол OpenFlow видаляє і додає підпотоки, повторюючи цю процедуру, тому графік а Рис.45 нагадує сходи. Використання пропускної здатності для обох протоколів обчислюється від початку експерименту в момент часу t = 10 і до t = 53 сек.[4]

2.3.3.Потік високого пріоритету для обох користувачів зі взаємним поділом

Користувачі А і Б мають високопріоритетний трафік і потребують для цього пропускну здатність. Контролер виділяє пропускну здатність, яка була присвоєна для трафіку з низьким пріоритетом, і розподіляє дані ресурси між обома користувачами, як показано на Рисунку 2.32. Коли вимоги щодо виділення смуги для високопріоритетного трафіку задоволені, контролер сигналізує про зміну потоків, щоб користувач Б міг використовувати дану смугу для низькопріоритетного трафіку.

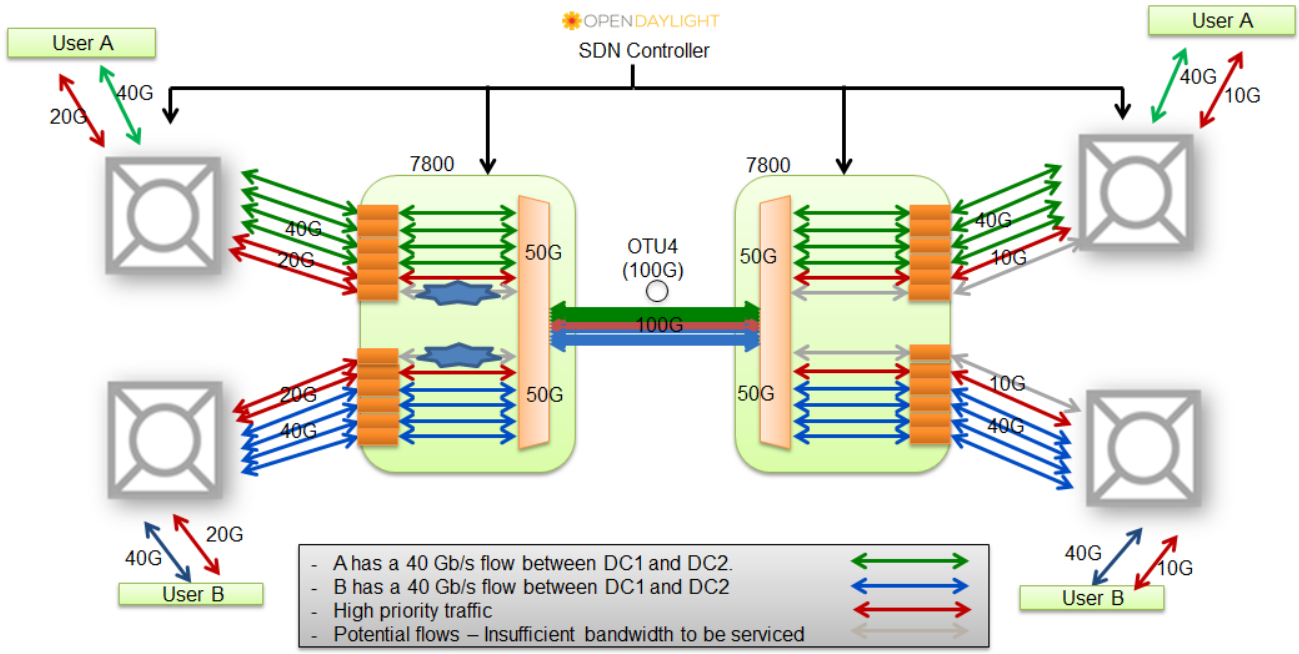

Рисунок 2.32. Потік високого пріоритету для обох користувачів зі взаємним поділом.

В даному сценарії обидва користувачі запрошують смугу пропускання шириною 20Гбіт/с для задоволення вимог щодо передачі високопріоритетного трафіку, але доступна лише смуга шириною 20Гбіт/с, яка використовується для низькопріоритетного трафіку. Контролер вирішує розділити доступну смугу, призначивши кожному користувачеві по 10Гбіт/с пропускної здатності для трафіку з високим пріоритетом, як це зображено в Таблиці 2.5. Немає жодних змін в потоці, який призначається для трафіку з високим пріоритетом користувача Б, оскільки канал, призначений для трафіку користувача Б з низьким пріоритетом, буде використовуватися потоком високого пріоритету. Для високопріоритетного трафіку користувача А виділяється частина смуги, яка була призначена для низькопріоритетного трафіку користувача Б. Тобто відбудеться модифікація лише одного потоку.

Таблиця 2.5. Виділення ресурсів для взаємного поділу каналів.

Користувач	**Ширина смуги**	**Тип потоку**	**Результат**
Користувач А	40Гбіт/с	Базовий потік	Забезпечено
	20Гбіт/с	Високий пріоритет	Забезпечено на 50% (10Гбіт/с)
Користувач Б	40Гбіт/с	Базовий потік	Забезпечено
	20Гбіт/с	Високий пріоритет	Забезпечено на 50% (10Гбіт/с)

На Рисунку 2.33 та Рисунку 2.34 зображено використання пропускної здатності протоколами NETCONF та OpenFlow. У момент часу t = 10 сек прикладний рівень надсилає менеджеру ресурсів запит на виділення пропускної здатності для високопріоритетного трафіку. В даному сценарії необхідно видалити та додати лише одне перехресне з’єднання. Протокол NETCONF потребує для цього 7 сек, а протокол OpenFlow - 8 сек. Протоколу NETCONF потрібно 0,5 сек, щоб видалити перехресне з'єднання і 3 сек, щоб додати нове. Протоколу OpenFlow для видалення перехресного з'єднання потрібно 0,75 сек і 3 сек, щоб додати нове. Через 8 секунд після цього система повертається до початкового стану. Протокол NETCONF і OpenFlow повертаються до початкового стану, шляхом видалення одного підпотоку з високим пріоритетом і додавання одного підпотоку з низьким пріоритетом. Протоколи NETCONF та OpenFlow потребують відповідно 7 сек та 8 сек, щоб досягти початкового стану.

Протокол NETCONF використовує смугу пропускання на 95,62%, а протокол OpenFlow - на 95,7%. Базуючись на результатах проведених дослідів видно, що чим менша кількість потоків, які необхідно змінити, тим більша величина використання пропускної здатності для обох протоколів.[4]

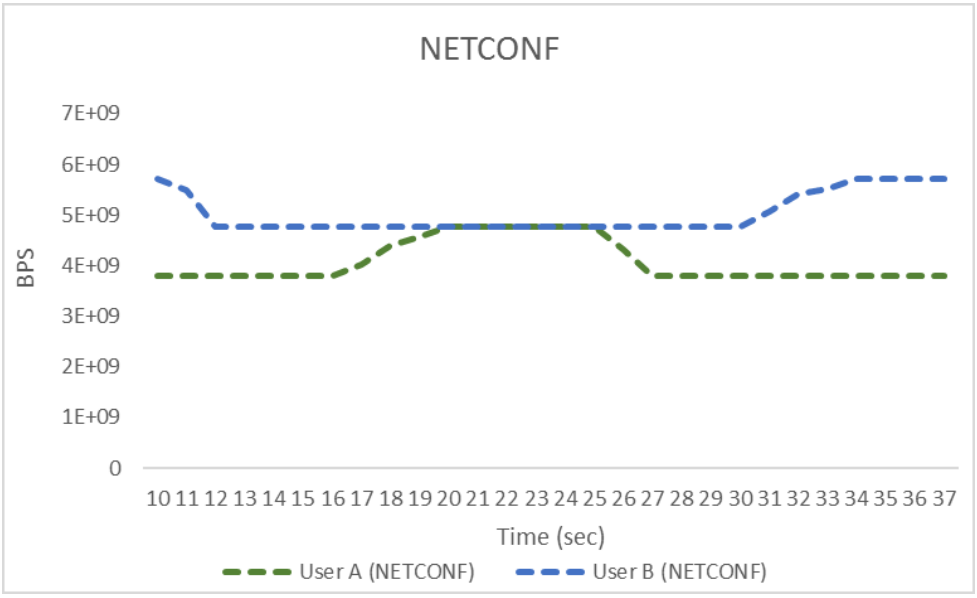

Рисунок 2.33. Використання пропускної здатності протоколом NETCONF при взаємному поділі смуги.

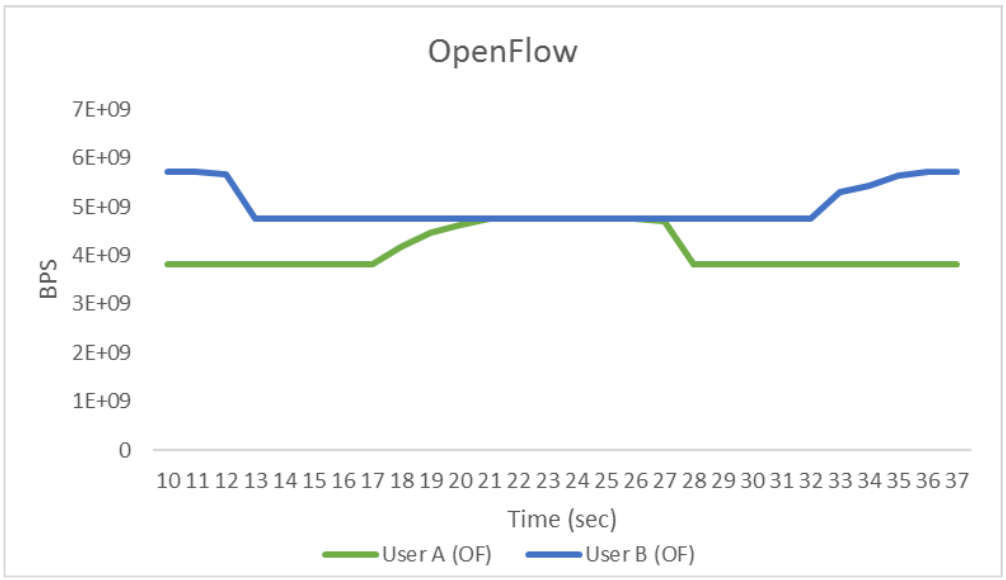

Рисунок 2.34. Використання пропускної здатності протоколом OpenFlow при взаємному поділі смуги.

2.3.4.Запит на смугу шириною 40 Гбіт/с

Даний сценарій зосереджений на оцінку роботи «Південних» протоколів при зміні великої кількості підпотоків. Користувач А потребує додаткову смугу шириною 40Гбіт/с. Контролер SDN забирає смугу з низькопріоритетного потоку та з кількох базових підпотоків користувача Б, щоб задовольнити вимоги користувача А, як показано на Рисунку 2.35.

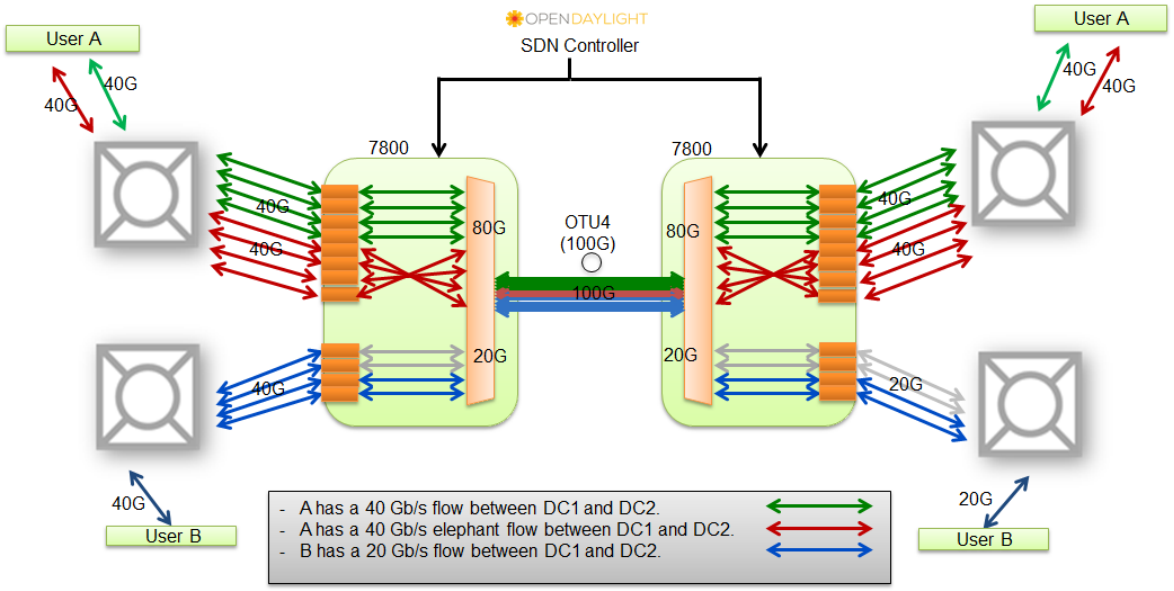

Рисунок 2.35. Запит користувача А на смугу шириною 40 Гбіт/с.

Після задоволення вимог користувача А система повертається до початкового стану. Якщо інший користувач (користувач Б) також вимагає смуги пропускання шириною 40Гбіт/с, то йому необхідно почекати поки запит користувача А на виділення смуги буде виконано.

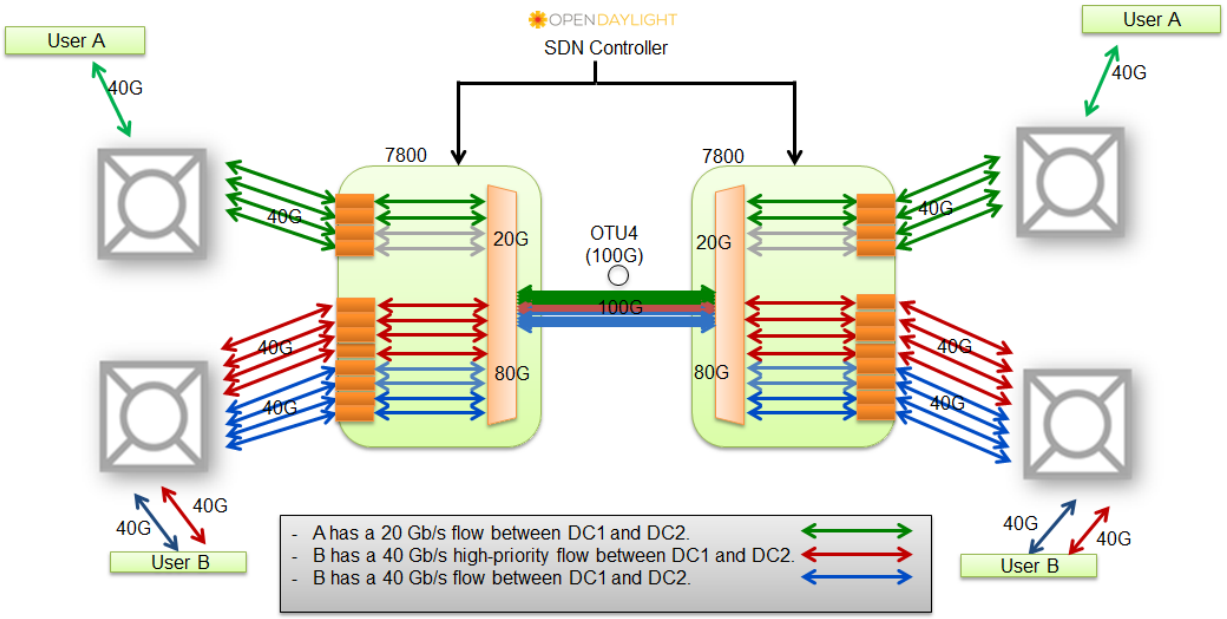

Рисунок 2.36. Запит користувача Б на смугу шириною 40 Гбіт/с.

На Рисунку 2.36 зображено підпотоки для випадку, коли користувач Б запрошує додаткову смугу шириною 40Гбіт/с. На Рисунку 2.37 та Рисунку 2.38

зображено етапи виконання запиту на виділення смуги пропускання за допомогою протоколів NETCONF та OpenFlow відповідно. Користувач взаємодіє з додатком BoD, який надсилає запит менеджеру ресурсів. Система перебуває в початковому стані з 10 встановленими підпотоками. Користувач А запрошує смугу пропускання шириною 40Гбіт/с і в результаті видаляються та додаються 4 нові перехресні з'єднання. Запит користувача А обслуговується протягом 8 секунд. Після закінчення 8 секунд менеджер ресурсів отримує запит на виділення смуги пропусканння шириною 40Гбіт/с для користувача Б. Після цього видаляються 6 існуючих перехресних з'єднань та додають 6 нових перехресних з'єднань для обслуговування запиту користувача Б. Через 8 секунд, коли система повернеться до початкового стану, 2 перехресні з'єднання видаляються та додаються 2 нових.

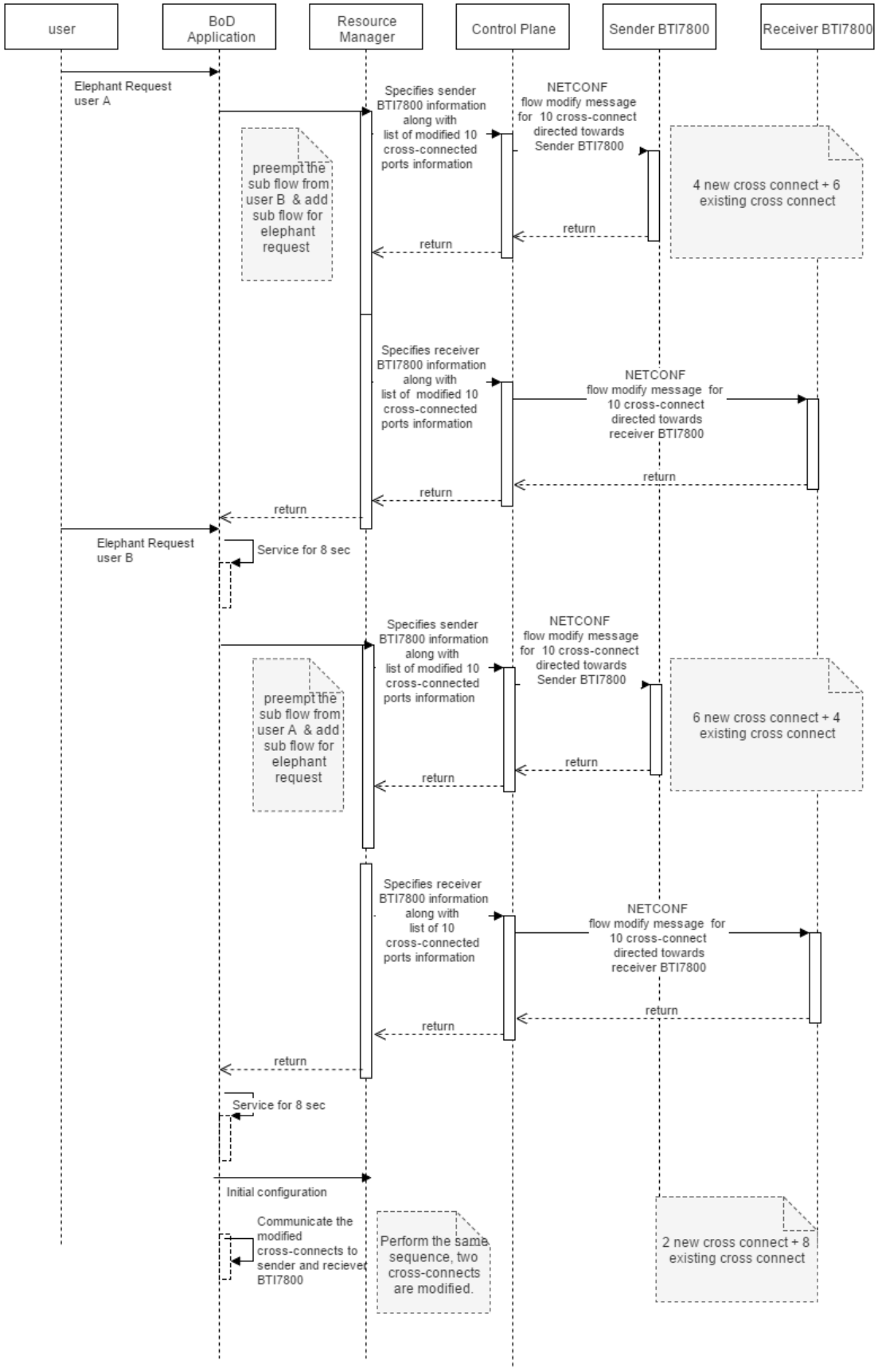

Рисунок 2.37.Етапи виконання запиту на виділення смугиза допомогою NETCONF.

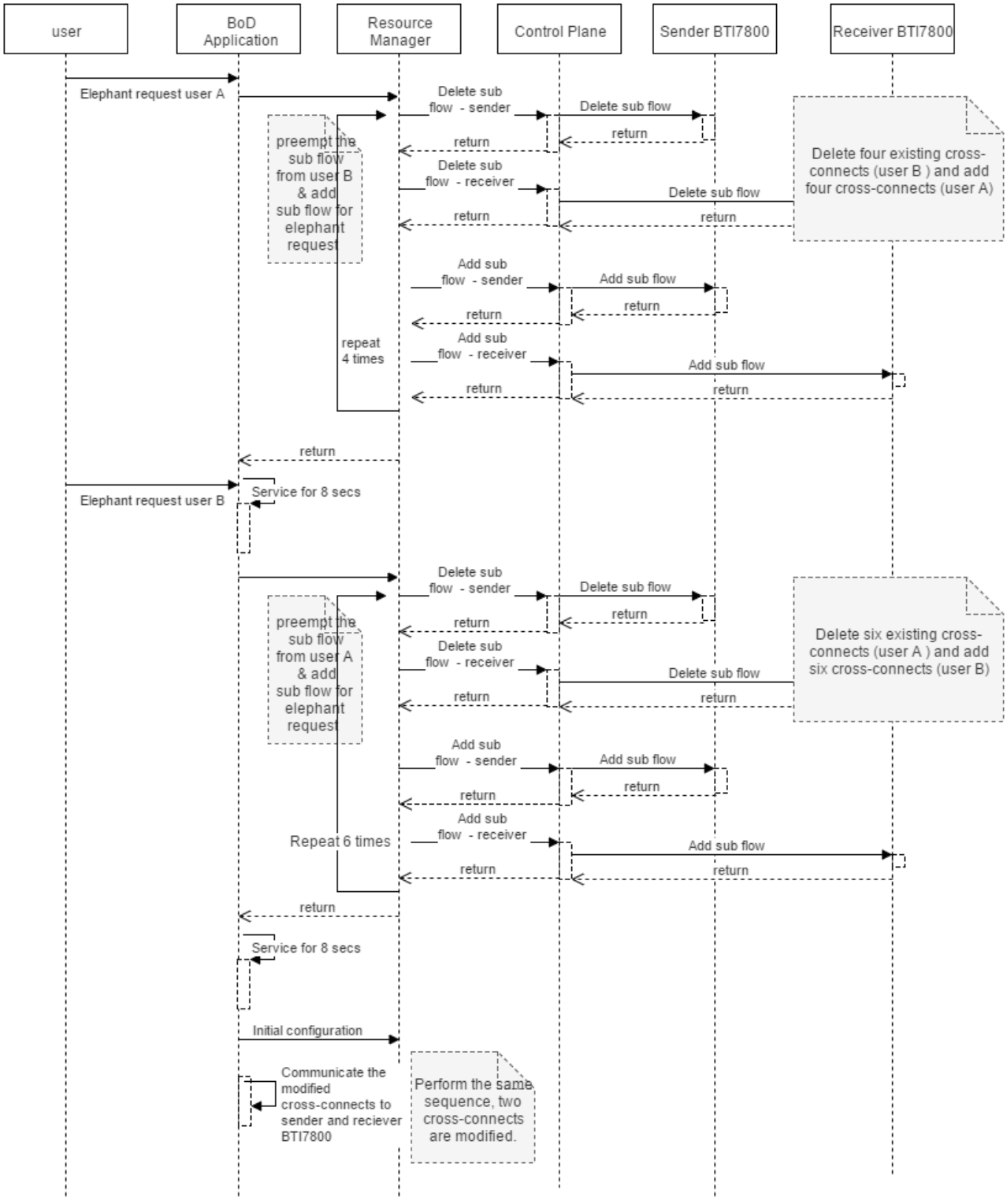

Рисунок 2.38. Етапи виконання запиту на виділення смугиза допомогою OpenFlow.

Результати роботи протоколів NETCONF та OpenFlow зображені на Рисунку 2.39 та Рисунку 2.40 відповідно. Дослідження починається в момент часу t = 10 сек, а система перебуває в початковому стані. В момент часу t = 10 сек прикладний рівень сигналізує менеджеру ресурсів про запит на виділення

смуги. Необхідно видалити 4 перехресні з'єднання і додати 4 нових в обох центрах обробки даних. Дана операція завершена в момент часу t = 36 сек і t = 40 сек для протоколів NETCONF і OpenFlow відповідно. На забезпечення даного запиту користувача А протоколам NETCONF та OpenFlow необхідно 26 сек та 30 сек відповідно. Протокол NETCONF потребує 13 сек для обслуговування одного центру обробки даних, 3 сек, щоб додати одне перехресне з'єднання та 1 сек для видалення всіх необхідних перехресних з'єднань. Протокол OpenFlow потребує 3 сек, щоб додати перехресне з'єднання, і 0,75 сек для видалення перехресного з'єднання. Через 8 секунд надходить запит від користувача Б на виділення смуги 40Гбіт/с. Протоколи NETCONF і OpenFlow передають дану інформацію в моменти часу t = 45 сек і t = 49 сек. Загальна кількість підпотоків, які необхідно змінити, становить шість, що включає в себе видалення шести перехресних з'єднань і додавання шести перехресних з'єднань. Протоколи NETCONF і OpenFlow потребують для цього 38 сек і 45 сек відповідно, при t = 83 сек і t = 94 сек. Система готова забезпечувати смугу 40Гбіт/с для користувача Б. Як було вказано раніше, NETCONF потребує 3 сек для додавання одного перехресного з'єднання і 1 сек для видалення всіх перехресних з’єднань, що в сумі становить 18 сек для додавання шести перехресних з'єднань, і 1 сек для видалення всіх перехресних з'єднань в одному центрі обробки даних. OpenFlow потребує 3 сек для додавання перехресного з'єднання і 0,75 сек для видалення перехресного з'єднання, що в сумі становить 18 сек для додавання шести перехресних з'єднань і 4,5 сек для видалення шести перехресних з'єднань в одному центрі обробки даних. Система перебуває в даному стані протягом 8 секунд, перш ніж повернеться до початкового стану. NETCONF досягає початкового стану в момент часу t = 104 сек, що займає 13 секунд. OpenFlow досягає початкового стану в момент часу t = 117 сек, що займає 15 секунд. Як видно з результату дослідження, кількість підпотоків, які потрібно видалити, не впливає на продуктивність роботи протоколу NETCONF. У протоколі OpenFlow кожен підпотік, який потрібно видалити, супроводжується окремим повідомленням і тому це займає більше часу.

Використання пропускної здатності обчислюється за період часу від t = 10 сек до t = 124 сек. NETCONF використовує пропускну здатність на 77,83%, а OpenFlow - на 92,92%. З одного боку, той факт, що OpenFlow обробляє кожен підпотік в окремому порядку, виступає як перевага, оскільки ресурси каналу постійно використовуються. З іншого боку, той самий факт виступає як недолік, оскільки чим більше підпотоків потрібно змінити, тим повільніше працює OpenFlow і тим більше повідомлень передається між контролером та елементами мережі. Протокол NETCONF швидше обробляє запити, але при цьому ресурси каналу не використовуютьься протягом періоду часу, пропорційного кількості потоків, які потрібно налаштувати. [4]

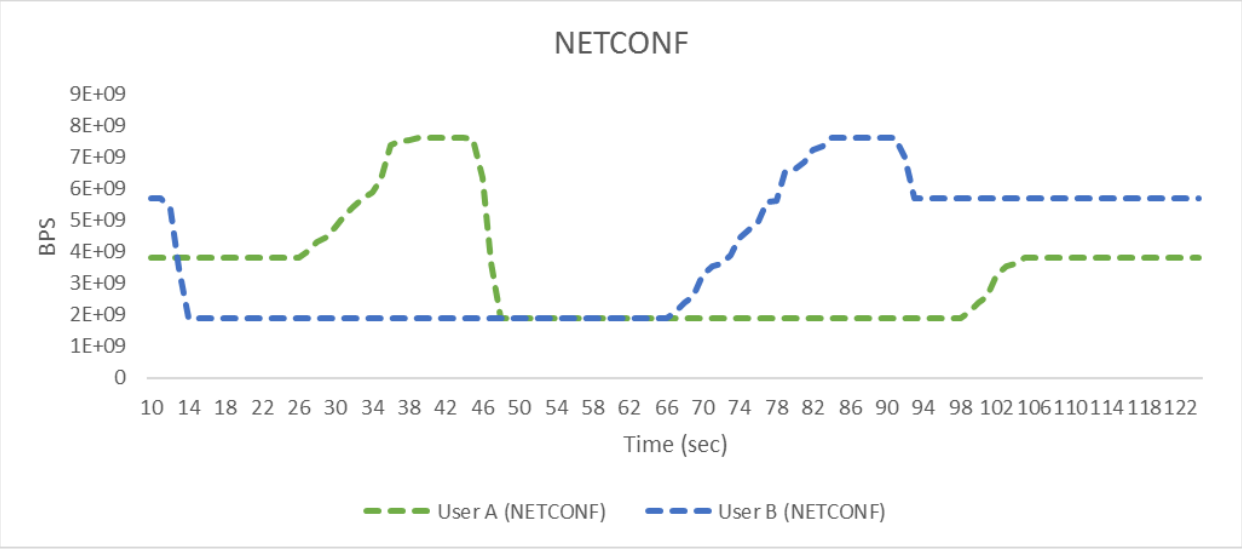

Рисунок 2.39. Виділення смуги шириною 40Гбіт/с протоколом NETCONF.

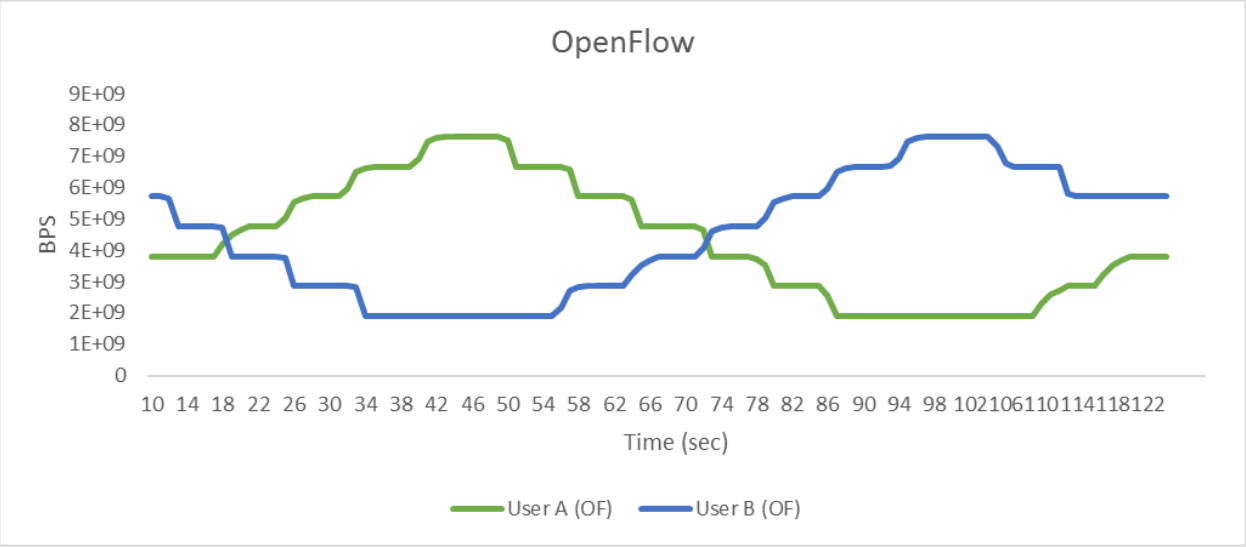

Рисунок 2.40.Виділення смуги шириною 40Гбіт/с протоколом OpenFlow.

2.3.5.Терміновий запит на смугу шириною 100 Гбіт/с

Якщо користувач А надсилає терміновий запит на виділення смуги шириною 100Гбіт/с, то контролер SDN забере ресурси, виділені для користувача Б, і виділить їх для задоволення вимог користувача А (Рисунок 2.41).

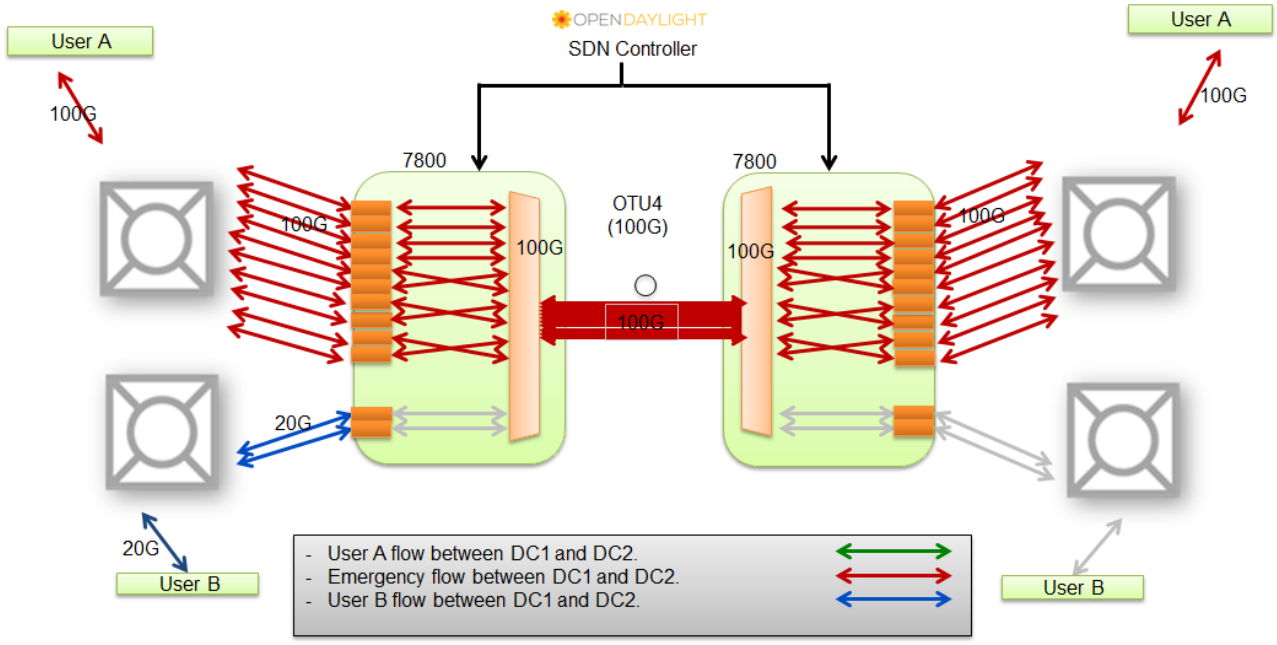

Рисунок 2.41. Терміновий запит користувача А на смугу шириною 100 Гбіт/с.

Після виконання термінового запиту користувача А система повертається до початкового стану. Якщо користувач Б відправляє аналогічний терміновий запит, то контролер SDN виділяє необхідні потоки для задоволення даної вимоги, як показано на Рисунку 2.42.

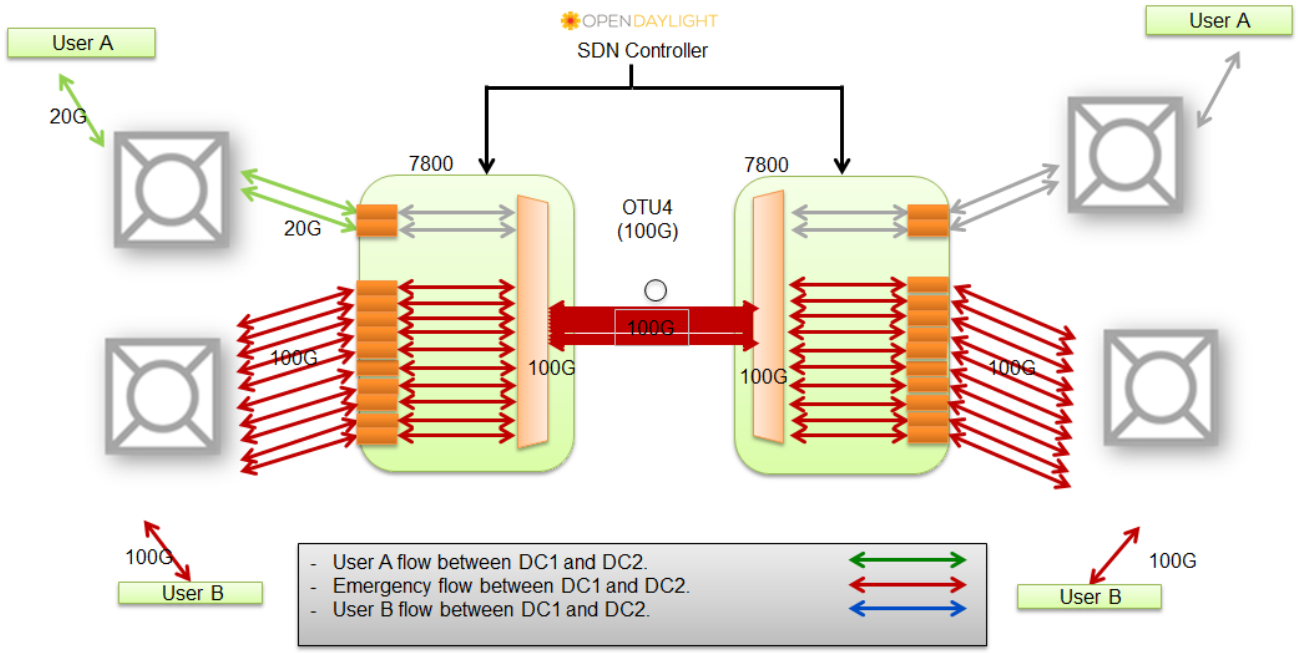

Рисунок 2.42 Терміновий запит користувача Б на смугу шириною 100 Гбіт/с.

Результати роботи протоколів NETCONF та OpenFlow зображені на Рисунку 2.43 та Рисунку 2.44 відповідно. Дослідження починається в момент t = 10 сек, а система перебуває в початковому стані. В момент часу t = 10 сек прикладний рівень надсилає менеджеру ресурсів терміновий запит на виділення смуги. Для виконання даного запиту необхідно видалити і додати 6 перехресних з'єднань. Дана операція завершена в момент часу t = 48 сек і t = 55 сек для протоколів NETCONF і OpenFlow відповідно. Протоколи NETCONF і OpenFlow потребують дял цього 38 сек і 45 сек відповідно. Через 8 секунд надходить повідомлення про необхідність забезпечення смуги для користувача Б. Протоколи NETCONF і OpenFlow передають дану інформацію в момент часу t = 55 сек і t = 65 відповідно. Загальна кількість змінених підпотоків дорівнює десяти. Протоколи NETCONF і OpenFlow потребують 65 сек і 76 сек відповідно і в моменти часу t = 120 сек і t = 140 сек терміновий запит користувача Б буде виконаний. Через 8 сек система повертається до початкового стану. Протокол NETCONF досягає початкового стану в момент часу t = 154 сек, тобто потребує 26 сек. Протокол OpenFlow досягає початкового стану в момент часу t = 179 сек, тобто потребує 30 сек. Загальна кількість додаткових підпотоків, які потрібно змінити дорівнює чотирьом.

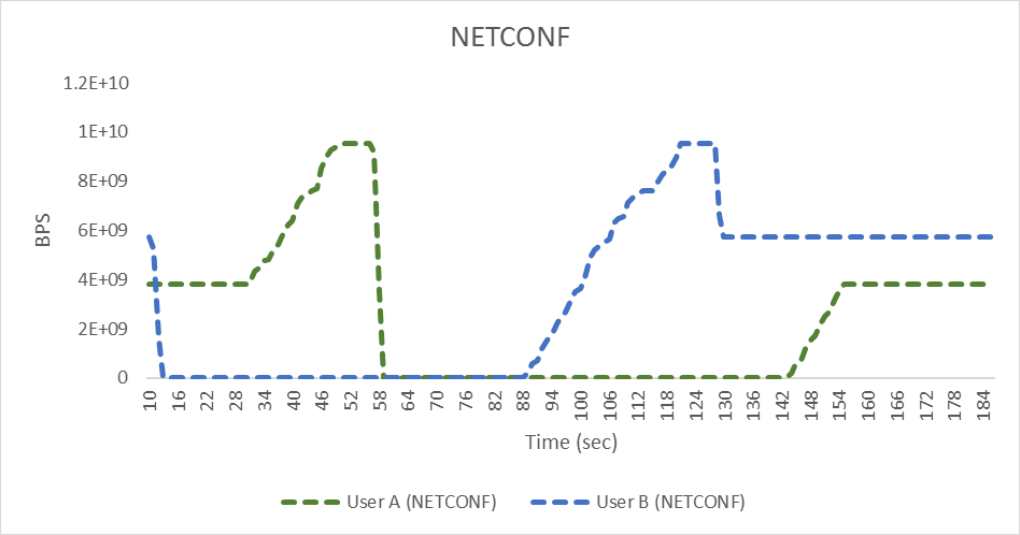

Рисунок 2.43.Виділення смуги шириною 100Гбіт/с протоколом NETCONF.

Використання пропускної здатності обчислюється за період часу, коли t = 10 сек і до t = 186 сек. Протокол NETCONF в даному сценарії використовує пропускну здатність на 59,55%, а OpenFlow - на 92,92%. Як видно з результатів дослідження - протокол NETCONF швидше обробляє запити, але ресурси каналу не використовуються протягом періоду, пропорційного кількості потоків, які потрібно налаштувати.[4]

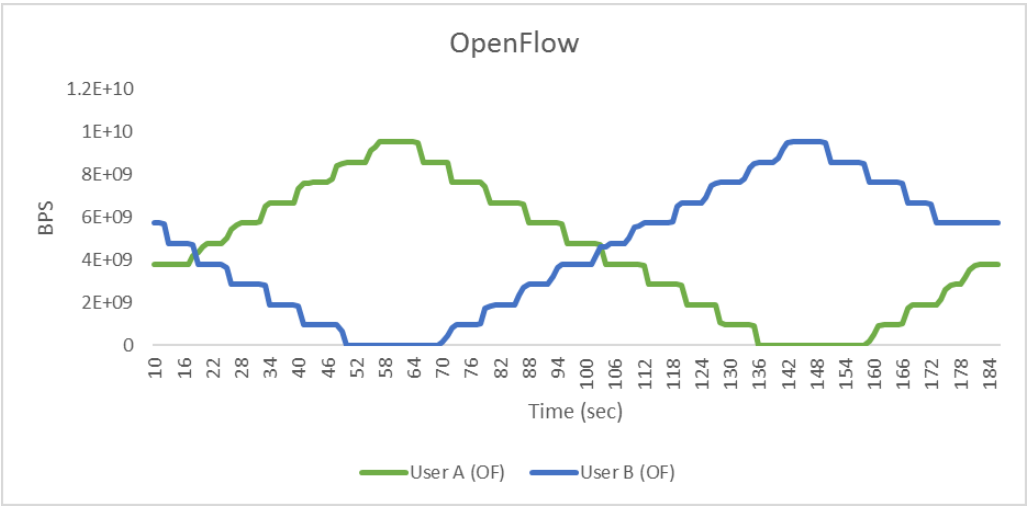

Рисунок 2.44.Виділення смуги шириною 100Гбіт/с протоколом OpenFlow.

2.3.6.Стрес-тест

Суть даного тесту полягає в тому, що два різних додатки одночасно налаштовують певний пристрій за допомогою технології SDN. Обидві програми "А" та "Б" намагатимуться одночасно змінити конфігурацію пристрою. Операційні дані пристрою зберігаються в операційній базі даних, а запити на зміну конфігурації, які надсилає додаток, зберігаються в конфігураційній базі даних.

2.3.6.1.NETCONF

Протокол NETCONF працює на основі моделі YANG. Коли додаток "А" надсилає запит на модифікацію, база даних конфігурації контролера отримує операційний стан разом із запитом на модифікацію та зберігає дану інформацію в базі даних конфігурації (Config A), як зображено на Рисунку 2.45. Перед тим, як запит на модифікацію від додатку "А" буде успішно застосовано на пристрої, додаток "Б" надсилає свій запит на модифікацію, але дана інформація зберігається в іншому екземплярі конфігураційної бази даних (Config B). Після цього існуватиме одна операційна база даних та два різні екземпляри конфігураційної бази даних для даного пристрою.

Припустимо, що запит на модифікацію від додатку «А» першим надійшов до площини управління. Контролер за допомогою протоколу NETCONF обробляє дані з конфігураційної бази даних «Config A». Якщо зміни вдало застосовано, то пристрій надсилає повідослення NETCONF <edit-config> про успішність операції. Операційна база даних оновлюється, а конфігураційна база даних «Config A» видаляється. Додаток “Б” надсилає запит про зміну конфігурації і дана інформація зберігається в конфігураційній базі даних «Config B». Тепер протокол NETCONF намагається обробити дані з конфігураційної бази даних «Config B», але оскільки її вміст було оновлено, то протокол NETCONF не може обробити запит.

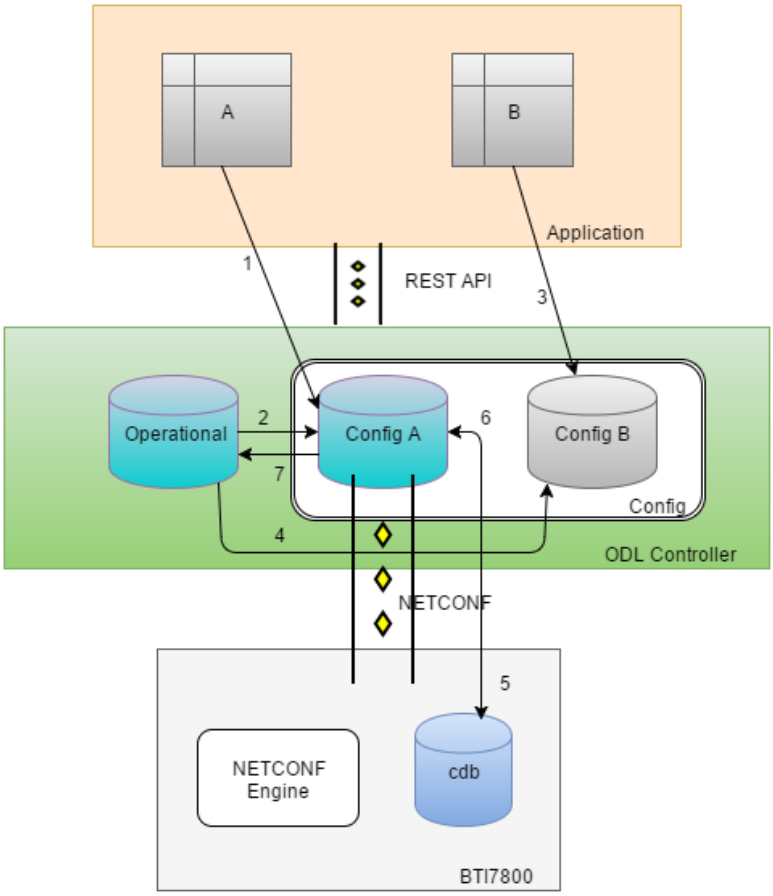

Рисунок 2.45. Обробка одночасних запитів на зміну конфігурації від двох додатків протоколом NETCONF.

Це призводить до того, що контролер видає помилку «NETCONF Connector Exception Error». Тобто контролер не може обробляти одночасні запити на модифікацію одного пристрою.[4]

2.3.6.2.OpenFlow

Процес обробки одночасних запитів на зміну конфігурації від двох додатків протоколом OpenFlow зображено на Рисунку 2.46. На відміну від протоколу NETCONF, OpenFlow не працює за моделлю даних і обробляє інформацію наступним чином. Контролер SDN транслює вхідну інформацію у вигляді повідомлень про модифікацію потоку OpenFlow. Конфігураційна база даних зберігає всю інформацію про додавання чи модифікацію правил потоків. Коли контролер отримує запити на зміну конфігурації одночасно від обох додатків А і Б, то зберігає їх в окремій конфігураційній базі даних. Обидві команди на модифікацію потоку передаються одночасно до пристрою BTI7800

і він одночасно їх отримує. Клієнт OpenFlow не може одночасно обробити два запити на модифікацію потоку від контролера, тому BTI7800 видає помилку.[4]

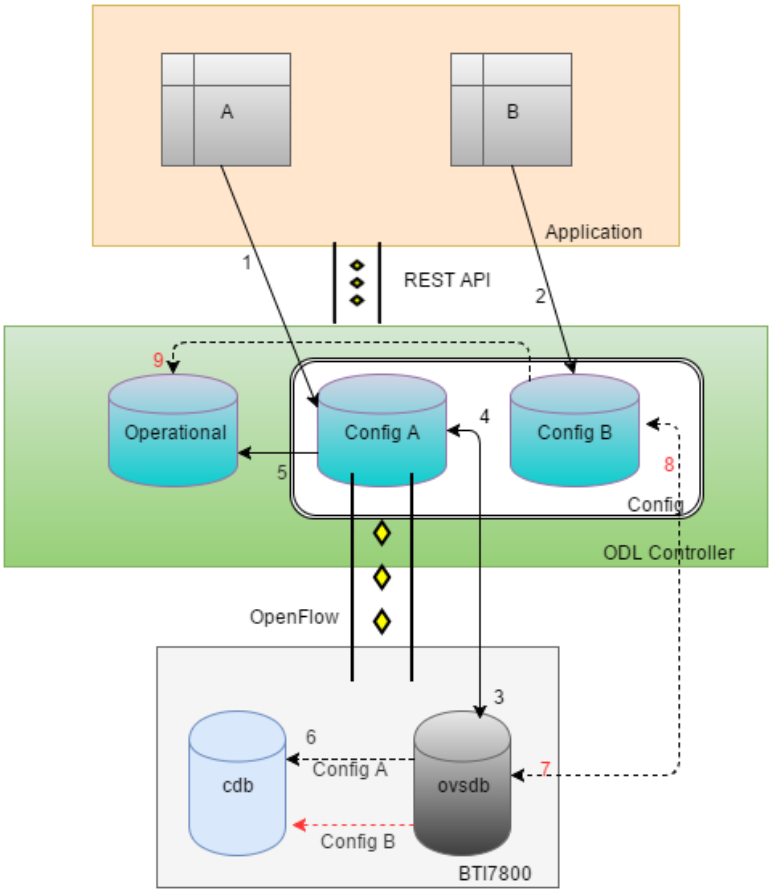

Рисунок 2.46. Обробка одночасних запитів на зміну конфігурації від двох додатків протоколом OpenFlow.

Висновки до розділу 2

Отже, протоколи NETCONF та OpenFlow обидва забезпечують взаємодію між контролером та мережевими елементами, але мають відмінності в структурі та логіці роботи. Протокол NETCONF є конфігураційним протоколом, який не вносить зміни в процес обробки пакетів мережевим елементом, а протокол OpenFlow змінює логіку обробки пакетів мережевим елементом за допомогою використання таблиці потоків. Протокол NETCONF можна легко адаптувати до будь-якої мережевої архітектури за допомогою гнучкого набору можливостей. Розробники можуть створювати додаткові можливості, тобто протокол NETCONF можна адаптувати під пропрієтарний функціонал конкретного виробника. Протокол OpenFlow, в

свою чергу, вимагає стандартизовану архітектуру мережевого елементу і не може реалізувати пропрієтарний функціонал конкретного виробника. Також протокол OpenFlow не потребує складних та високовартісних мережевих елементів, які мають велику обчислювальну здатність, що сприяє зменшенню витрат на мережеве обладнання.

Важливими для порівняння даних протоколів є результати дослідження їх роботи в різних сценаріях використання. На Рисунку 2.47 представлено час, необхідний протоколам для зміни певної кількості перехресних з'єднань. На Рисунках 2.48, 2.49, 2.50 представлено результати порівняння протоколів NETCONF та OpenFlow для всіх сценаріїв, описаних в даній роботі.

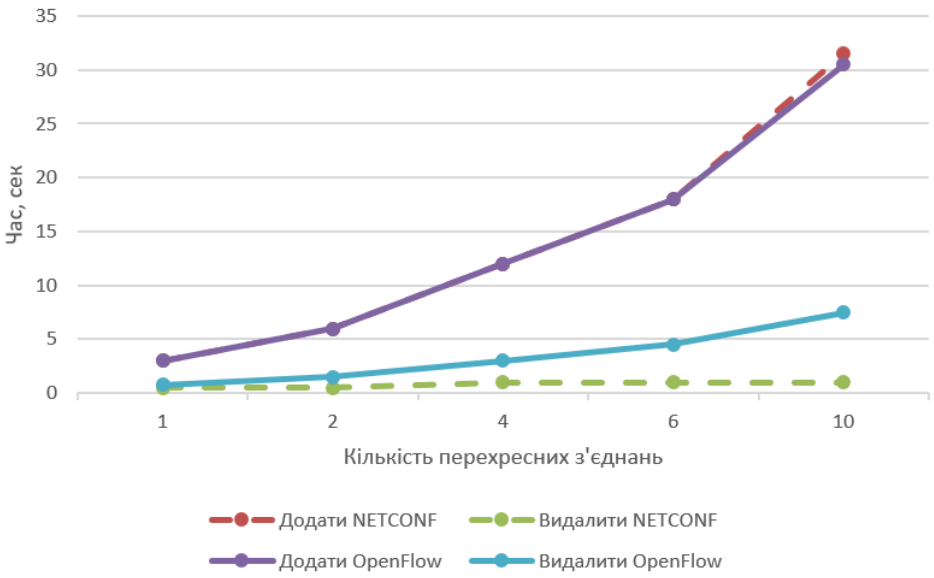

Рисунок 2.47. Час, необхідний протоколам для зміни конкретної кількості перехресних з'єднань.

Як видно з Рисунків, збільшення кількості перехресних з'єднань, вплине на продуктивність протоколу OpenFlow, оскільки контрольні повідомлення та час, необхідний для обробки кожного повідомлення, збільшуються пропорційно кількості перехресних з'єднань, які потрібно змінити. Для протоколу NETCONF незалежно від кількості перехресних з'єднань, буде відправлено одне повідомлення, а час необхідний для обробки буде пропорційним кількості потоків, які потрібно змінити.

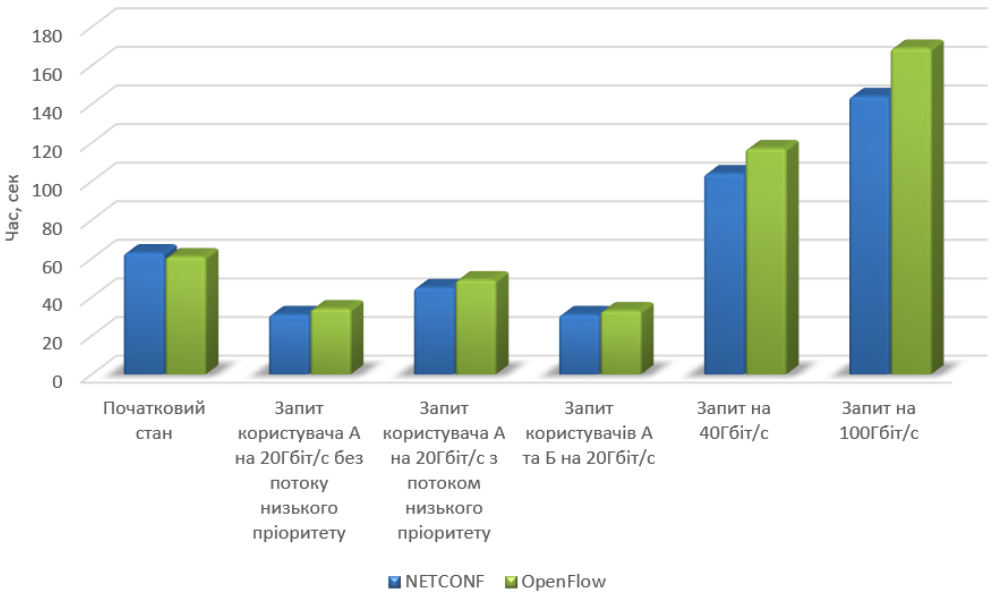

Рисунок 2.48. Час, необхідний для виконання вказаного набору конфігурацій обома протоколами.

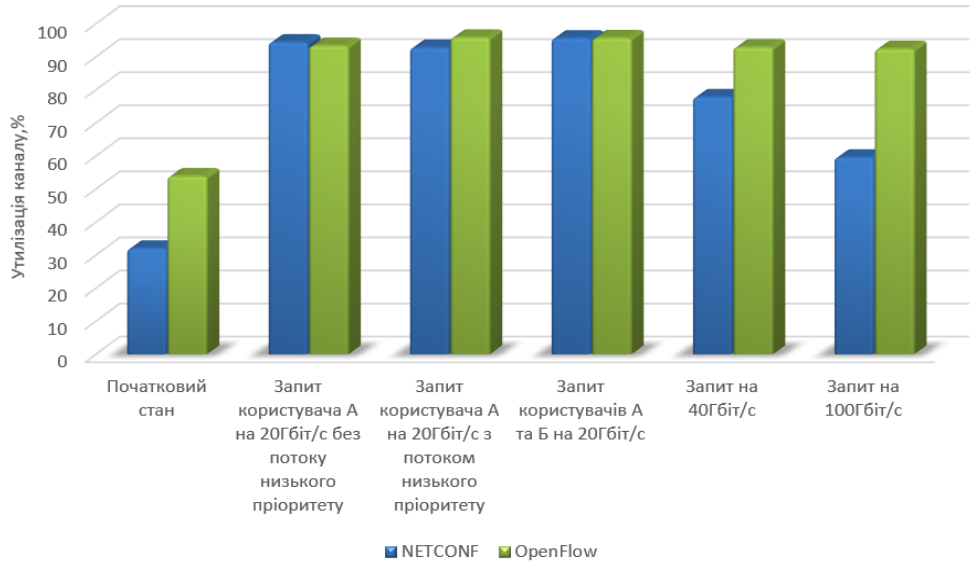

Рисунок 2.49. Величина використання пропускної здатності.

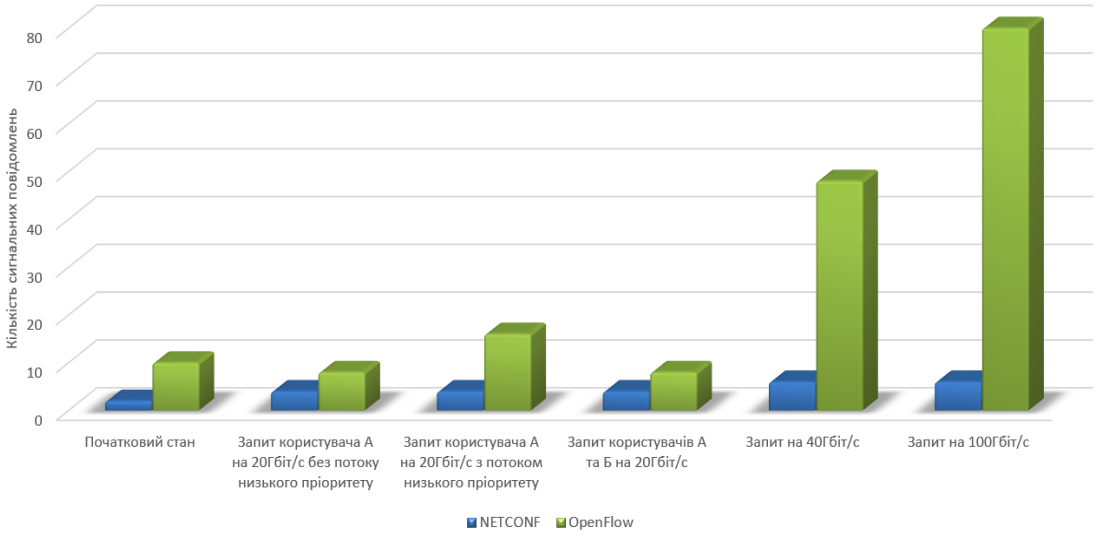

Рисунок 2.50. Кількість сигнальних повідомлень для реалізації конкретного сценарію.

Отже, за результатами дослідження протокол NETCONF швидший, ефективніший по кількості контрольних повідомлень, що передаються між контролером та мережевим елементом. З іншого боку, протокол OpenFlow обробляє кожне перехресне з'єднання окремо і таким чином, він ефективно використовує пропускну здатність, але він повільніший, ніж NETCONF і кількість контрольних повідомлень для нього збільшується пропорційно кількості правил потоку, які необхідно модифікувати. У випадку, коли невелике простоювання каналу допустиме для мережі - NETCONF є кращим протоколом. OpenFlow є кращим протоколом для випадку, коли простоювання каналу недопустиме.

3.ПРИНЦИПИ ПОБУДОВИ КОНТРОЛЕРА.ФУНКЦІОНАЛЬНА СХЕМА

SDN є комплексною системою взаємодії елементів як логічної, так і фізичної природи. Застосування контролера як єдиної інтелектуальної точки управління дозволяє значно спростити логіку роботи і вартість обладнання SDN архітектури, так як зникає необхідність у підтримці та обробці безлічі стандартів і протоколів управління на транспортному рівні. Загальний фреймфорк архітектури SDN зображена на Рисунку 3.1.

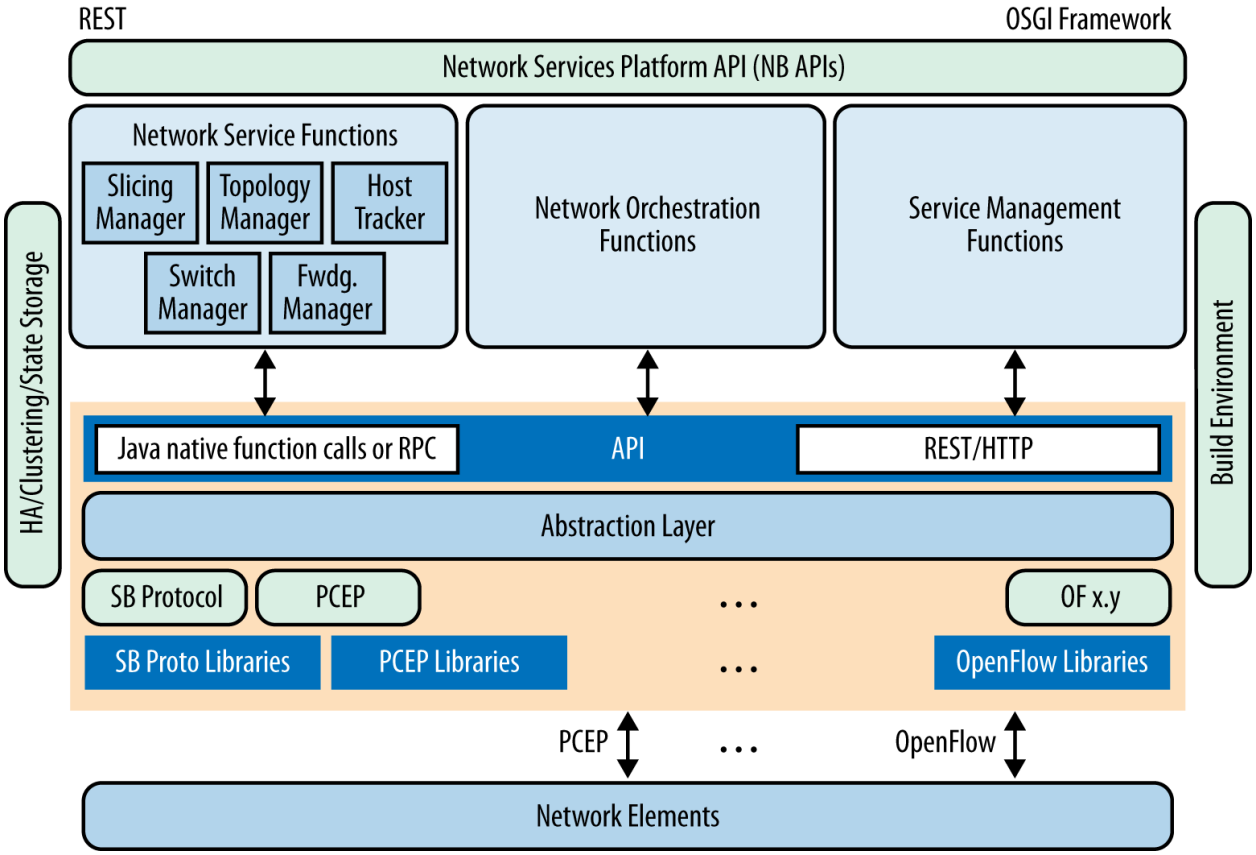

Рисунок 3.1. Загальний фреймфорк архітектури SDN.

Контролер SDN можна охарактеризувати як програмну систему або набір систем, які разом забезпечують:

1. управління станом мережі, яке може включати необхідність використання баз даних. Ці бази даних служать сховищем для інформації, отриманої від керованих мережевих елементів та відповідного програмного забезпечення, а також інформації від додатків SDN, яка включає стан мережі, певну конфігураційну інформацію, вивчену топологію та інформацію про сеанси

управління. У деяких випадках контролер може мати кілька цільових процесів управління даними (наприклад, реляційні та нереляційні бази даних);

2. модель даних високого рівня, яка фіксує зв’язки між керованими ресурсами, політиками та іншими сервісами, що надаються контролером. У багатьох випадках ця модель даних будується на базі мови моделювання YANG;
3. сучасні API, які роблять можливою взаємодію контролера з додатками SDN. У деяких випадках контролер та його API є частиною розробленого середовища, яке генерує API-код з моделі, яка використовується. Деякі системи забезпечують гнучкі середовища розробки, які дозволяють розширювати основні можливості та використовувати API для нових модулів, у тому числі для таких, що підтримують динамічне розширення можливостей контролера:
    i. безпечні TCP-сесії між контролером та агентами мережевих елементів;
    ii. стандартизований протокол для підтримки стану мережі;
    iii. механізм моніторингу пристроїв, топології та сервісів, а також інші сервіси, орієнтовані на мережу або ресурси.[7]

3.1.Контролер OpenDayLight

Контролер SDN виступає в якості пункту управління в мережі SDN, який передає інформацію комутаторам/маршрутизаторам "вниз" через SBI та додаткам "вгору" через NBI. Платформа контролера SDN зазвичай складається з набору модулів, які виконують різні мережеві завдання. Деякі основні завдання включають в себе інвентаризацію пристроїв у мережі, визначення їх можливостей та збір їх статистики. Також можуть бути додані певні розширення для підвищення функціональності та підтримки вдосконалених функцій, таких як оркестрування нових правил по всій мережі, віртуалізація мережевих функцій та аналітика мережевого трафіку. Існує багато варіантів

реалізації SDN-контролерів, але в даній роботі виеористано контролер, який задовольняє наступним вимогам:

- контролер SDN має відкритий вихідний код;
- контролер підтримує протоколи NETCONF і OpenFlow;
- контролер має добре розвинені «Північні» протоколи;
- взаємодія користувача з контролером відбувається через графічний інтерфейс.

Проект ODL - це платформа з відкритим кодом для SDN. Він використовує відкриті протоколи для забезпечення централізованого, програмного управління та моніторингу мережевих пристроїв. Як і багато інших контролерів SDN, ODL підтримує OpenFlow, а також пропонує готові до встановлення мережеві рішення в рамках своєї платформи. Фреймворк ODL зображено на Рисунку 3.2.

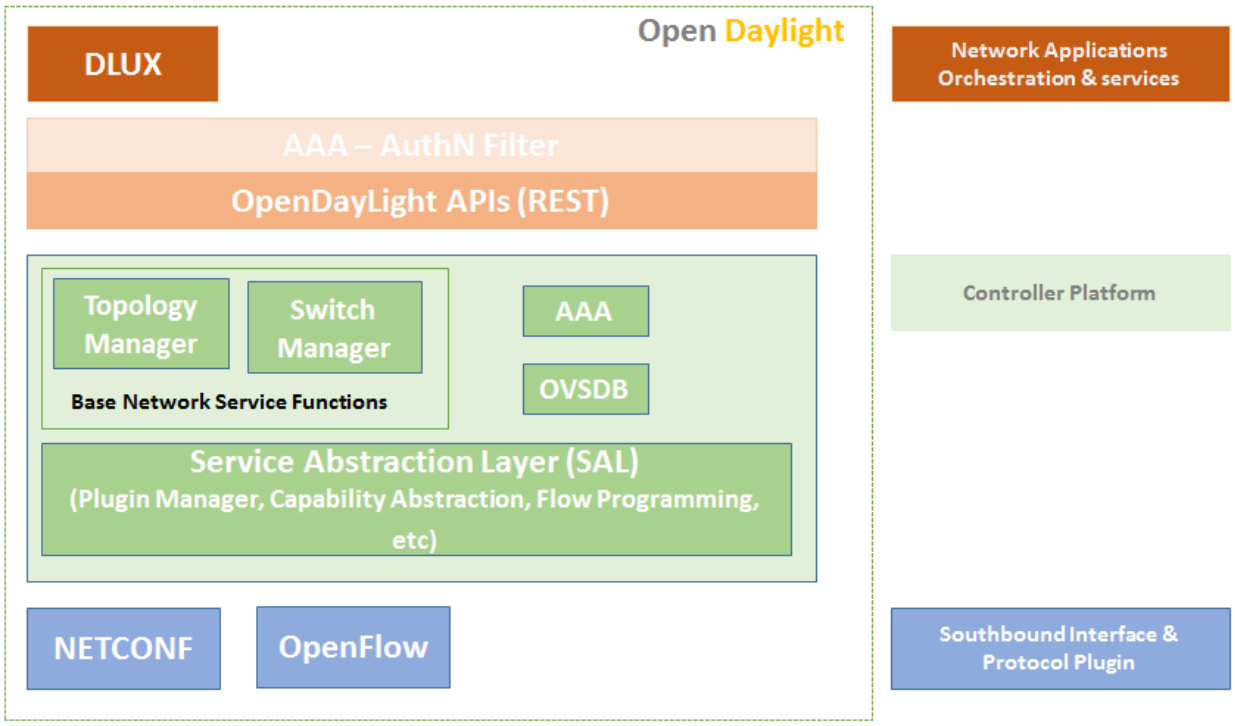

Рисунок 3.2. Фреймворк OpenDayLight.

Контролер має наступні характеристики:

- він підтримує архітектуру мікросервісів, в якій мікросервіс - це певний протокол або послуга, яку користувач може активувати в рамках мережі SDN. Наприклад, плагін забезпечує зв'язок між площиною управління та пристроями мережі за допомогою реалізованих «Південних» протоколів або пропонуються такі сервіси як AAA чи підтримка функції комутації;
- різні мережеві послуги та оркестрація підтримуються на програмному рівні. Контролер має веб-інтерфейс DLUX, який забезпечує сучасний графічний інтерфейс для взаємодії з контролером, налаштування та адміністрування;
- REST - це архітектурний стиль WWW, який спілкується за допомогою протоколу HTTP. Тобто використовує ті ж самі команди HTTP: GET, POST, PUT, DELETE, які веб-браузери використовують для пошуку веб-сторінок та взаємодії з віддаленими серверами. Контролер ODL підтримує REST в якості «Північного» протоколу. REST використовує RPC-повідомлення для зв'язку з площиною управління контролера;
- контролер підтримує архітектуру MD-SAL, яка дозволяє налаштувати додаткові сервіси або плагіни;
- основний рівень SAL складається з баз даних, які поділяються на конфігураційні та операційні. Конфігураційна база даних містить інформацію про конфігурацію і дозволяє користувачам змінювати її через REST API. Друга база даних містить операційну інформацію, яка надходить із системи, і показує користувачам успішність застосування їх конфігурацій.

Платформа контролера складається з набору модулів, як зображено на Рисунку 3.3. Контролер спілкується з мережевою інфраструктурою за допомогою плагінів «Південного» протоколу та надає базові мережеві сервіси за допомогою набору менеджерів, які формують групу «Base Network Service Functions» і включаються в себе Topology Manager, Stats Manager, Switch Manager, Forwarding Rules Manager та Host Tracker.

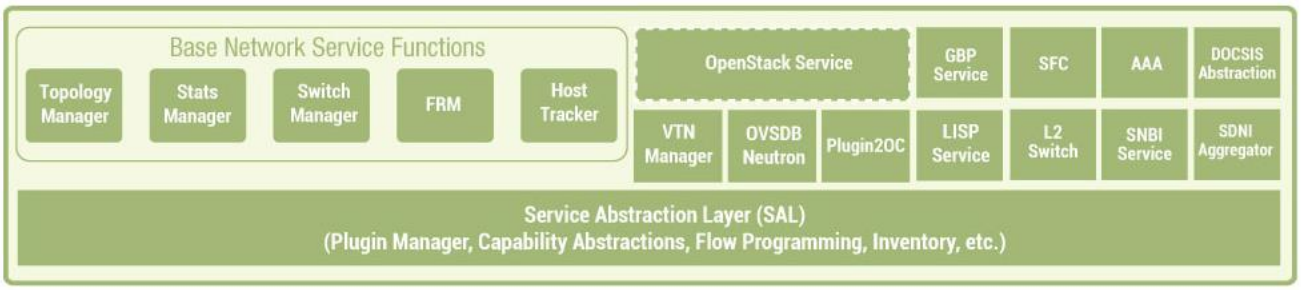

Рисунок 3.3.Функціональна схема контролеру OpenDayLight.

Topology Manager - зберігає та обробляє інформацію про керовані мережеві пристрої. Коли контролер запускається, Topology Manager створює кореневий вузол в операційному піддереві топології. Потім він прослуховує сповіщення та оновлює це піддерево певними деталями топології, включаючи всі досліджені комутатори та їх взаємозв'язки. Сповіщення від інших компонентів, таких як Switch Manager або Device Manager, також можуть містити інформацію про топологію.

Statistics Manager - реалізує збір статистики, надсилаючи запити на всі керовані комутатори та зберігає відповіді в операційному піддереві статистики.

Switch Manager - надає інформацію про мережеві вузли та стан їх портів. Як тільки контролер виявить мережевий елемент, його параметри будуть збережені у дереві даних Switch Manager.

Forwarding Rules Manager - керує основними правилами переадресації, вирішує їх конфлікти та підтверджує їх. Forwarding Rules Manager спілкується з плагінами «Південного» протоколу та завантажує правила в керовані комутатори.

Inventory Manager - запитує та оновлює інформацію про комутатори та порти, якими керує OpenDaylight, гарантуючи, що база даних інвентаризації точна та актуальна.

Host Tracker - зберігає інформацію про кінцеві хости (дані канального рівня, тип комутатора, тип порту, дані мережевого рівня) та забезпечує API,

які отримують інформацію про кінцевий вузол. Host Tracker може працювати в статичному або динамічному режимі. В динамічному режимі Host Tracker використовує ARP для відстеження стану бази даних. У статичному режимі база даних заповнюється вручну через «Північні» API.

Крім вищезгаданих основних служб, існує ряд інших служб, які знаходиться на тому ж архітектурному рівні, взаємодіють з основними модулями контролера та надають конкретні функції для відповідних служб та модулів.

Одним з найбільш важливих елементів контролера OpenDaylight є рівень SAL, який з'єднує плагіни протоколу та модулі Service Network Function Modules.

Model-Driven SAL забезпечує загальний підхід до розробки плагінів, що дозволяє об'єднувати «Північні» та «Південні» API з структурами даних, що використовуються в різних компонентах контролера. У MD-SAL всі пов'язані зі статусом дані зберігаються у вигляді моделі об'єкта документа (DOM), відомої як «дерево даних». В платформі контролера OpenDaylight використовуються наступні два типи дерев даних:

- операційне дерево DOM, яке модулі контролерів використовують для зберігання певної тимчасової інформації;
- дерево конфігурації DOM, яке використовується для зберігання поточного стану контролера.

MD-SAL використовує YANG, як мову моделювання для опису всіх мережевих служб. YANG дозволяє легко розробити будь-який базовий модуль, прикладний додаток чи компонент платформи, що забезпечує вбудовані структури та типи, наприклад контейнери, списки та листи, в той час як розробник може вказати будь-які додаткові структури для типів даних певних завдань. Після визначення моделей YANG компілятор видає відповідні інтерфейси Java, а наступним кроком є реалізація цих автоматично створених інтерфейсів Java.[9]

3.2.Контролер OpenFlow

Архітектура контролера OpenFlow складається з декількох рівнів, кожен з яких відповідає за ряд необхідних функціональних можливостей.

Основними рівнями OpenFlow контролера є:

- рівень взаємодії з мережею;
- рівень обробки OpenFlow повідомлень;
- рівень обробки подій;
- рівень мережевих сервісів і внутрішніх додатків;
- інтерфейс для мережевих додатків контролера;
- рівень мережевих додатків.

За своєю суттю контролер виступає в ролі елемента реагування: отримує повідомлення від комутаторів по каналах управління і виробляє відгуки, які змінюють вміст таблиць комутації.

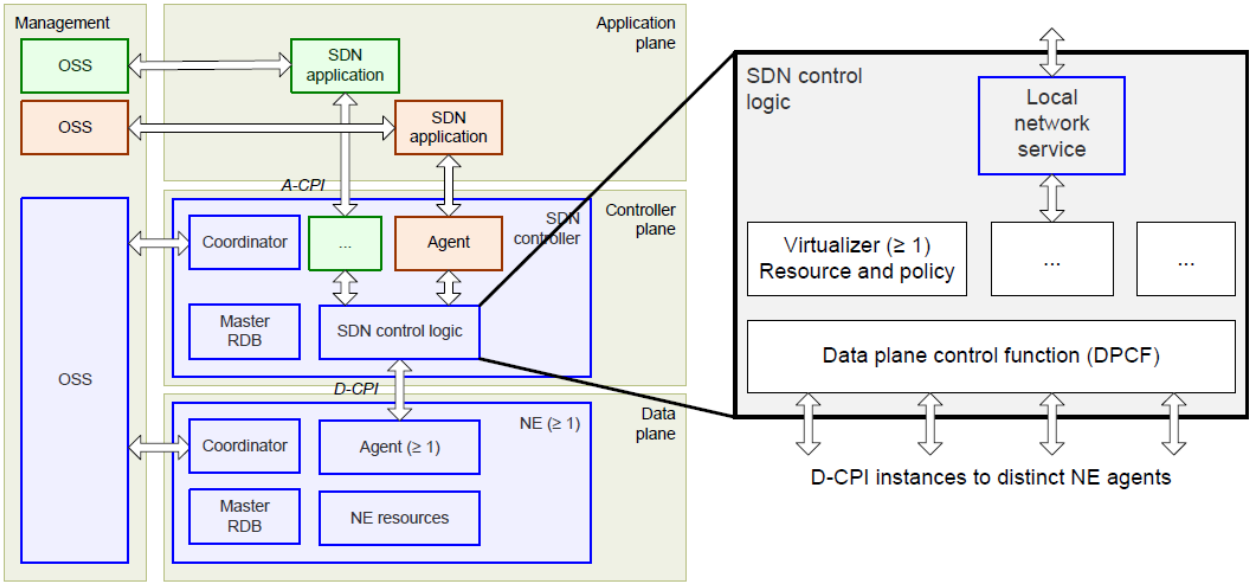

Рисунок 3.4. Функціональна схема SDN-контролера OpenFlow.

Зображена на Рисунку 3.4 архітектура не визначає внутрішній дизайн контролера SDN. Контроль SDN логічно централізований. Контролер зазвичай має множину підмереж, що охоплює більше одного фізичного мережевого елемента. Контролер SDN являється власником віртуальних ресурсів,

виділених йому управлінням. Додатковими функціями SDN контролера, залежно від обставин, можуть виступати наступні:

- знання топології та обчислення шляху (контролер також може викликати зовнішню службу для цих функцій);
- створення та підтримка додаткової абстрактної моделі ресурсів для своїх програм, з ресурсами, обмеженими примусовою політикою.

SDN-контролер повинен координувати ряд взаємопов'язаних ресурсів, часто розподілених між низкою підрядних платформ, а іноді забезпечувати транзакційну цілісність, це називається оркестровкою.

Архітектура SDN не визначає внутрішній дизайн або реалізацію контролера SDN. Вона може бути єдиним монолітним процесом; може бути конфедерацією ідентичних процесів, призначених для спільного завантаження або захисту один одного від поломок; може бути сукупністю окремих функціональних компонентів у спільній організації; може використовувати зовнішні служби для деяких своїх функцій, наприклад, для обчислення шляху. Допускається будь-яка комбінація цих альтернатив: контролер SDN розглядається як чорний ящик. Компоненти контролера можуть вільно виконуватись на довільних обчислювальних платформах, включаючи обчислення ресурсів локально для фізичного мережевого елементу. Вони також можуть виконуватись на розподілених та, можливо, міграційних ресурсах, таких як віртуальні машини в центрах обробки даних. Кілька компонентів контролера можуть мати спільний доступ до мережевих ресурсів, але щоб відповідати принципам SDN, вони також повинні:

- бути налаштованими для управління незв’язаними наборами ресурсів або дій;
- бути синхронізованими один з одним, щоб вони ніколи не видавали суперечливі команди.

Такі проблеми, як завантаження, синхронізація, міграція, резервне копіювання, аудит, керування випуском програмного забезпечення

контролера тощо, є внутрішніми для чорного ящика контролера SDN, і не стосуються архітектури SDN.

Контролер SDN є чорним ящиком, але необхідно концептувати мінімальний набір функціональних компонентів в контролері SDN, а саме: функція керування площиною даних (DPCF), координатор, віртуалізатор та агент. Залежно від вимог логічної централізації, контролер SDN може включати довільні додаткові функції.

Компонент DPCF ефективно володіє доступними йому підлеглими ресурсами та використовує їх відповідно до інструкцій координатора або віртуалізатора, які їх контролюють. Ці ресурси набувають форми екземпляра інформаційної моделі, доступ до якої здійснюється через агента на підпорядкованому рівні. Оскільки масштаб контролера SDN повинен охоплювати декілька мережевих елементів або навіть декілька віртуальних мереж (з окремим екземпляром D-CPI для кожного), DPCF має включати функцію, яка працює для агрегації. Ця функція називається оркестровою. Проте, наведена архітектура не визначає оркестровку як окремий функціональний компонент.

Для налаштування клієнтських і серверних середовищ потрібна функція керування. Координатор є функціональним компонентом контролера SDN, який діє в якості менеджера. Клієнти та сервери вимагають управління з усіх точок зору на рівні даних, контролю та прикладних програм, тому функціональні блоки координаторів є повсюдними.

В архітектурі SDN віртуалізація - це розподіл абстрактних ресурсів конкретним клієнтам або додаткам; в NFV метою є абстрагування мережевих функцій від виділеного обладнання, наприклад, щоб вони могли бути розміщені на серверних платформах у центрах хмарних даних. Функціональний об'єкт, який підтримує екземпляр інформаційної моделі та політику на A-CPI (application-controller plane interface), називається віртуалізатором. Віртуалізатор є екземпляром інформаційної моделі для

кожної клієнтської програми або організації. Координатор виділяє ресурси, які використовує віртуалізатор для представлення A-CPI, який він видає своїм клієнтам, і він встановлює політику, яку віртуалізатор застосовує. Результатом цих операцій являється створення агента для даного клієнта. Віртуалізатор може розглядатися як процес, який отримує запити клієнта через A-CPI, перевіряє запити щодо політики та ресурсів, призначених клієнту, перекладає запит на умови основних ресурсів і передає результати до DPCF та D-CPI. Віртуалізатор, DPCF та, можливо, інші функції контролера SDN повинні співпрацювати, щоб забезпечити такі функції, як інтерпретація запитів, спільне використання ресурсів та цілісність транзакцій.

Будь-який протокол повинен бути визначеним в певному функціональному об'єкті. Модель контролер-агент підходить для взаємозв'язку між контрольованим об’єктом та об’єктом, що контролює і застосовується рекурсивно до архітектури SDN. Контрольований об’єкт визначається агентом - функціональним компонентом, який представляє ресурси та можливості клієнта в середовищі сервера. Агент у даному контролері SDN на рівні N представляє ресурси та дії, доступні для клієнта або програми контролера SDN, на рівні N + 1. Агент на рівні даних N-1 представляє ресурси та дії, доступні для даного контролера SDN рівня N. Незважаючи на те, що фізичне місцезнаходження агента знаходиться всередині довірчого домену сервера, агент знаходиться в довірчому домені клієнта.

Щоб уникнути перевизначення, архітектура описує лише функції, необхідні для контролера SDN, але не виключає додаткових функцій. Вони можуть мати форму додатків або функцій, підтримуваних контролером. Ці функції можуть експортуватися до деяких або всіх клієнтів зовнішніх додатків сервера або використовуватись внутрішньо адміністратором провайдера для власних цілей. Як компоненти контролера SDN, такі програми або функції підлягають таким же вимогам синхронізації, що й інші компоненти контролера. Щоб полегшити інтеграцію з стороннім програмним

забезпеченням, інтерфейси до таких програм або функцій можуть бути такими самими, як і інші у A-CPI.[3]

3.3.Контролер Huawei

На Рисунку 3.5 зображена структурна схема контролера Huawei. Як видно з рисунку, контролер будується на базі операційної системи Linux, а фреймворк на базі контролера OpenDayLight.

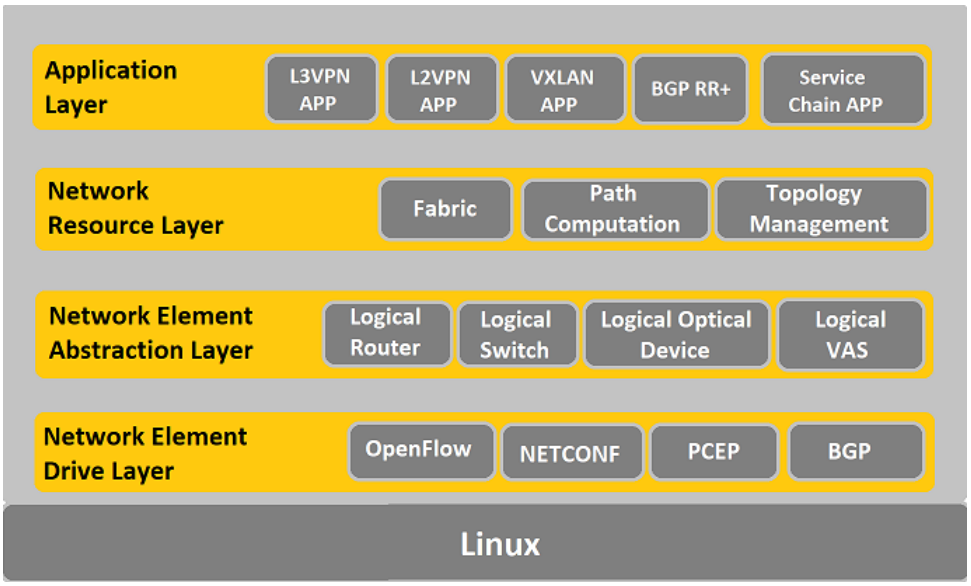

Рисунок 3.5.Структурна схема контролера Huawei.

Архітектура контролера складається з 4-х рівнів, а саме:

- Application Layer, який складається з багатьох сервісних додатків;
- Network Resource Layer, який складається з модулів Fabric, Path Computation та Topology Management, що відповідають за збір інформації про топологію мережі, управління ресурсами топології та обчислення сервісних шляхів;
- Network Element Abstraction Layer, який складається з елементів Logical Router, Logical Switch, Logical Optical Device та Logical VAS, що реалізують відповідний віртуальний пристрій;

- Network Element Drive Layer, який складається з чотирьох функціональних модулів для «Південних» інтерфейсів OpenFlow, NETCONF, PCEP та BGP. Дані модулі використовуються для зв’язку з мережевими елементами.

3.4.Контролер NetCracker

На Рисунку 3.6 зображено фреймворк SDN компанії NetCracker. Даний фреймворк базується на структурі контролера OpenFlow.

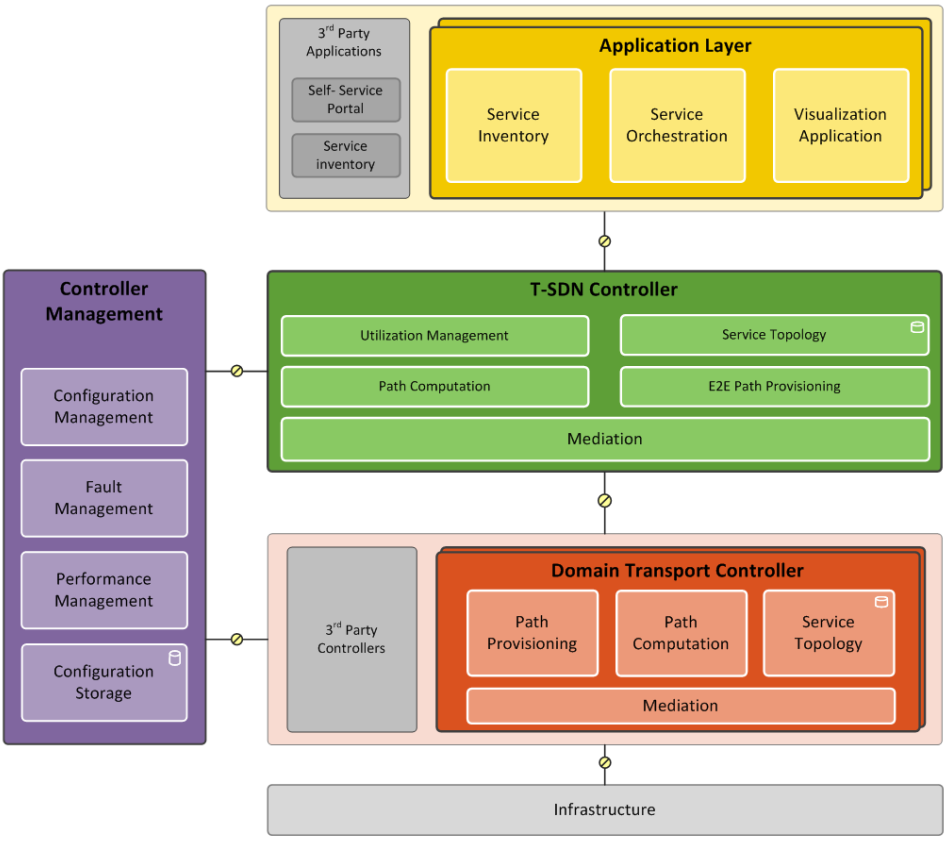

Рисунок 3.6. Фреймворк SDN компанії NetCracker.

Контролер складається з наступних основних модулів:

- Path Provisioning, який включає в себе операції побудови маршрутів на логічному рівні, а також безпосередньо взаємодіє з медіатором (Mediation), від якого отримує оброблену інформацію з підключених доменних контролерів;
- Path Computation, який відповідає за визначення маршруту на мережевому графі з урахуванням різних критеріїв;

- Utilization Management, який відповідає за оптимізацію використання пропускної здатності каналів;
- Service Topology, який здійснює зберігання актуальних і запланованих сервісів і їх параметрів;
- Mediation, який відповідає за роботу з API нижніх пристроїв і здійснює підтримку досить великого набору протоколів мережевого управління, таких як RESTCONF, NETCONF, SNMP, SSH, OpenFlow, а також HTTP/HTTPS;
- Controller Management, який відповідає моделі FCAPS стандарту ISO і включає в себе управління конфігурацією, відмовами і продуктивністю контролера.[8]

Висновки до розділу 3

В даному розділі було було розглянуто принципи побудови і роботи контролерів на базі двох фреймворків: OpenDayLight та OpenFlow. Дані контролери мають відмнінності в архітектурі, але найважливішими є відмінності в реалізованому «Південному» протоколі, який використовується для взаємодії з мережевими елементами. Контролер OpenDayLight використовує протокол NETCONF, а контролер OpenFlow використовує протокол OpenFlow. Також було розглянуто контролери від компанії Huawei, в основі якого закладений фреймворк OpenDayLight та NetCracker, в основі якого закладений фреймворк OpenFlow.

Базуючись на результатах дослідження в Розділі 2, можемо зробити висновки, що SDN продукція компанії Huawei спрямована на забезпечення швидкої взаємодії між контролером та елементами мережі, а SDN продукція компанії NetCracker жертвує швидкістю взаємодії, щоб краще використовувати наявну пропускну здатність мережі.

4.ІНТЕГРАЦІЯ АРХІТЕКТУРИ SDN В ІСНУЮЧУ ІНФРАСТРУКТУРУ ОПЕРАТОРА ЗВ'ЯЗКУ

Архітектуру SDN можна інтегрувати в існуючу інфраструктуру оператора, пропонуючи корпоративним клієнтам такі сервіси, як канал точка-точка, доступ до мережі Інтернет або доступ до певного центру обробки даних.

4.1.Канал точка-точка

Канал точка-точка дає можливість реалізувати зв'язок між двома сайтами користувача в приватній віртуальній мережі. Для цього між маршрутизаторами двох сайтів встановлюється Overlay VPN тунель, через який буде передаватись трафік (Рисунок 4.1).

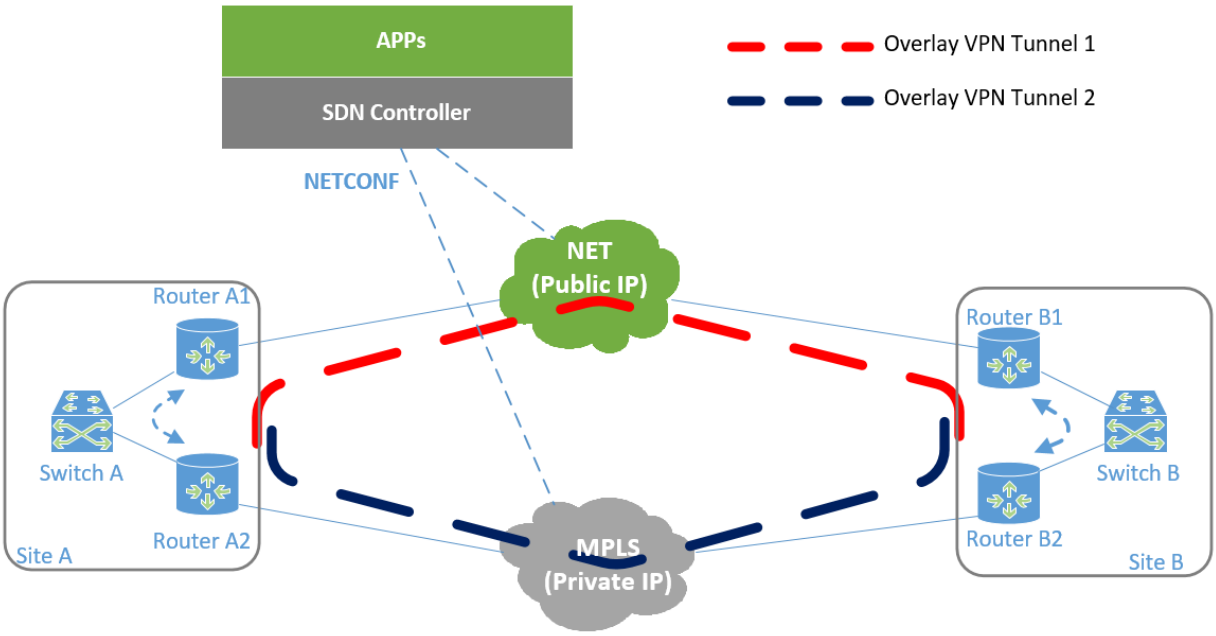

Рисунок 4.1. Реалізація каналу точка-точка між двома сайтами.

Overlay VPN тунель будується наступним чином:

- контролер визначає базову конфігурацію та налаштовує маршрутизатори через інтерфейс протоколу NETCONF;
- два Overlay VPN тунелі встановлюються між маршрутизаторами поверх андерлейної транспортної мережі, один з яких працює через Інтернет, а інший через MPLS;

- різні сайти використовують різні IP-сегменти, які виділяються контролером. Маршрутизатори на кожному сайті працюють в якості DHCP-сервера для динамічного виділення IP-адрес користувачам LAN-мережі;
- протокол BGP використовує тунелі для обміну IP-префіксами між сайтами.

Використання SD-WAN для даного сценарію дозволяє динамічно управляти рішеннями по передачі пакетів даних, шляхом оцінки ефективності, політик та стану певного шляху. Контролер стежить за продуктивністю мережі (джиттер, затримки, втрати пакетів) та на базі отриманих даних приймає рішення про передачу трафіку по найкращому шляху. Для побудови тунелю через мережу Інтернет можна використовувати прооткол IPSec, щоб забезпечити захист конфіденційності даних, що передаються.

4.2.Доступ до мережі Інтернет

Доступ до мережі Інтернет через центральний сайт є одним з базових сервісів(Рисунок 4.2). Суть даного сервісу заключається в тому, що певний трафік необхідно передати на центральний сайт, на якому працює мережевий екран (Firewall) перед відправленням його до мережі Інтернет, а деякий трафік може бути відправлений в мережу Інтернет напряму з локального сайту.

Відбувається це наступним чином:

- за допомогою протоколу NETCONF контролер конфігурує статичний маршрут за замовчуванням на маршрутизаторі центрального сайту, який перенаправляє трафік на мережевий екран. На маршрутизаторах локальних сайтів контролер конфігурує два VRF: VRF1 та VRF2. VRF1 використовується для оверлейної мережі, а VRF2 – для андерлейної мережі Інтернет. Між даними VRF контролер конфігурує політику, яка дозволяє перенаправляти трафік з одного VRF в інший;
- маршрутизатор на центральному сайті анонсує даний статичний маршрут решті сайтів за допомогою протоколу динамічної маршрутизації через сконфігуровані тунелі;

- користувачі на сайтах отримують приватні IP-адреси та IP-адресу DNS-серверу від локальних маршрутизаторів. Трафік, який необхідно передати на центральний сайт, передається через оверлейний тунель, а трафік, який необхідно передати до мережі Інтернет, фільтрується сконфігурованою політикою, переходить у VRF2 і передається через андерлейний тунель.

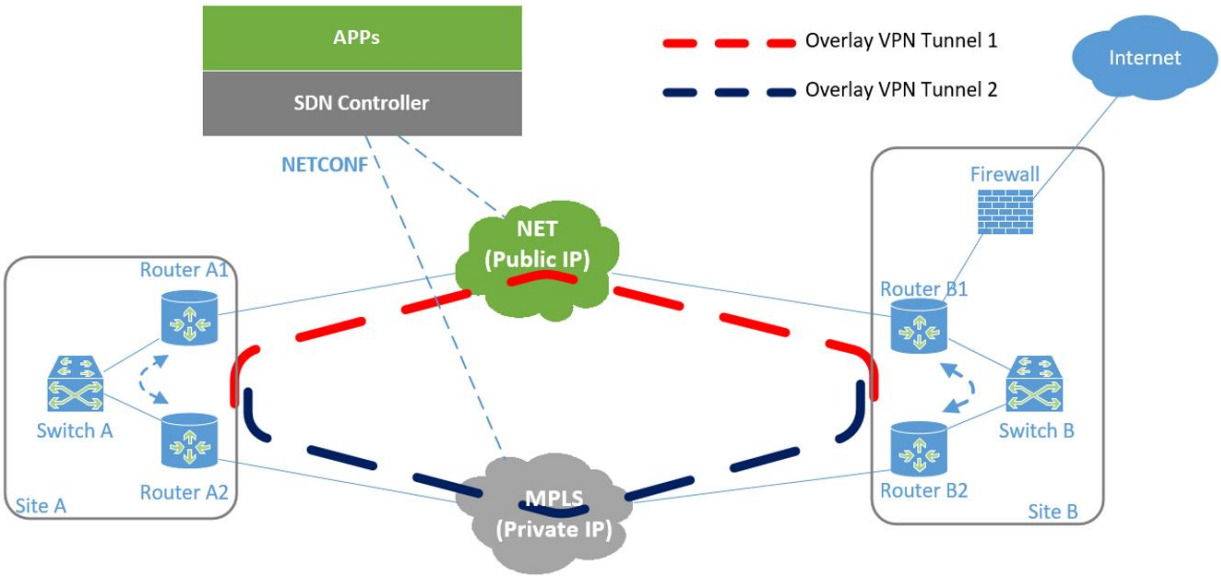

Рисунок 4.2. Реалізація доступу до мережі Інтернет.

4.3.Інтеграція до RAN-мережі оператора

Архітектуру SDN можна інтегрувати до RAN-мережі оператора зв’язку (Рисунок 4.3). Тоді архітектура RAN включатиме в себе наступні додаткові елементи:

- API, які впорядковують надання послуг та реалізують взаємодію користувача з мережею;
- контролер, який забезпечує взаємодію з API за допомогою Restful/https/YANG інтерфейсів, а взаємодія з елементами мережі відбувається за допомогою протоколу NETCONF. Також можлива інтеграція існуючих систем моніторингу, тоді контролер може збирати інформацію про мережеві елементи за допомогою Restful протоколів з даних систем моніторингу;

- NMS, яка збирає дані з елементів мережі за допомогою протоколу SNMP та може передавати їх контролеру за допомогою Restful протоколів.

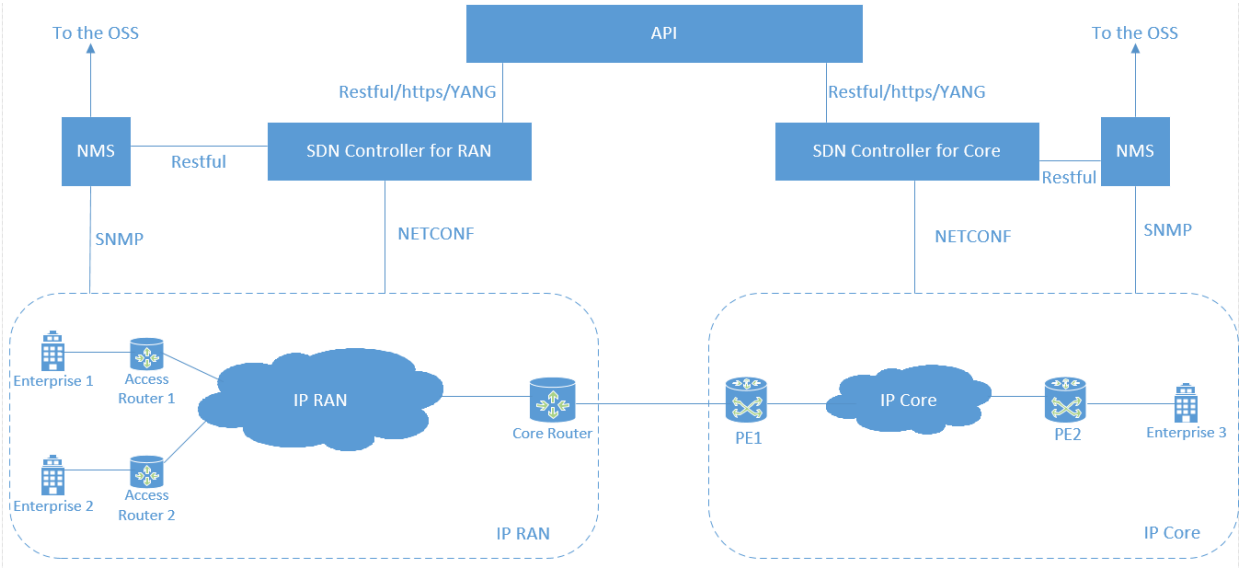

Рисунок 4.3. Інтеграція доRAN-мережі оператора

Висновки до розділу 4

В даному розділі було розглянуто варіанти інтеграції архітектури SDN в існуючу інфраструктуру оператора зв’язку з метою підвищення якості послуг, спрощення управління елементами мережі і прискорення процесу активації нових послуг для сценарію SD-WAN. Дана інтеграція не потребує кардинальних змін в структурі мережі та заміни існуючого мережевого обладнання.

ЗАГАЛЬНІ ВИСНОВКИ

Стрімке зростання обсягів трафіку, необхідність підтримки зростаючої кількості мобільних користувачів, формування високопродуктивних кластерів для обробки Великих Даних та віртуалізованих середовищ для надання хмарних сервісів - все це серйозно змінило вимоги до телекомунікаційних мереж. Все частіше традиційна мережа стає обмежуючим фактором розвитку обчислювальної інфраструктури.

В якості технології, яка може вирішити дані проблеми, все частіше використовується концепція SDN. Безліч IT організацій і мережевих провайдерів успішно використовують її, в першу чергу, з метою зниження вартості мережевої інфраструктури і витрат на її утримання, а також для забезпечення високого рівня керованості, захищеності, надійності мережі, для зменшення часу і складності розгортання нових сервісів. Згідно з концепцією SDN вся логіка управління переноситься на окремі централізовані пристрої - контролери. В основі підходу SDN лежить можливість динамічно управляти пересиланням даних по мережі за допомогою відкритих протоколів. Це дає можливість конфігурувати мережу як єдину систему, яка не залежить від виробника обладнання.

На даний момент існує багато варіантів реалізації концепції SDN для різних сценаріїв використання, тому перед споживачами даного продукту постає складне питання вибору необхідної архітектури SDN, яка задовольнятиме вимоги до конкретної мережі. Для того, щоб обрати підходящу архітектуру необхідно розуміти принципи роботи, побудови та відмінності у роботі різних реалізацій SDN. Ключовим моментом роботи мережі SDN є метод, згідно з яким контролер управляє елементами мережі, адже від нього залежить швидкість реакції елементів на команди контролера, ефективність використання наявних ресурсів мережі, складність інтеграції архітектури SDN з існуючою інфраструктурою оператора і навіть різноманітність доступного функціоналу.

В даній роботі було досліджено принципи роботи та характеристики двох найбільш популярних протоколів для управління мережевими пристроями: NETCONF та OpenFlow. За результатами дослідження протокол NETCONF швидший та ефективніший по кількості контрольних повідомлень, що передаються між контролером та елементами мережі. Протокол OpenFlow ефективно використовує наявну пропускну здатність, але він повільніший, ніж NETCONF. У випадку, коли невелике простоювання каналу допустиме для мережі - NETCONF є кращим протоколом. OpenFlow є кращим протоколом для випадку, коли простоювання каналу недопустиме. Дані результати дозволяють обрати протокол, який краще підходить для конкретного сценарію використання, базуючись на вимогах до мережі.

Базуючись на обраному «Південному» протоколі будуватиметься фреймворк SDN. Якщо протокол NETCONF більше підходить для задоволення вимог мережі, то необхідно використовувати фреймворк OpenDayLight, а якщо більш підходящим є протокол OpenFlow, то необхідно використовувати фреймворк OpenFlow.

В даній роботі приведено рішення для інтеграції ахрітектури SDN на базі фреймворку OpenDayLight в існуючу інфраструктуру оператора зв’язку, тобто для сценарію застосування SD-WAN, який потребує швидкої реакції елементів мережі на команди контролера, щоб динамічно управляти наявними ресурсами мережі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Scott Shenker UC Berkeley by 2011 [Електронний ресурс] – Режим доступу до ресурсу: https://inst.eecs.berkeley.edu/~ee122/fa11/notes/18-SDN122-lecture.pptx
2. Сетевые технологии SDN – Software Defined Networking [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/muk/blog/251959/
3. ONF, Software-Defined Networking: SDN architecture. ONF White Paper, 2014.
4. K. Muthukumar. Analysis of OpenFlow and NETCONF as SBIs in an SDN Environment, 2016
5. Network Configuration Protocol (NETCONF), IETF, Request for Comments: 6241, Obsoletes: 4741, Category: Standards, ISSN: 2070-1721
6. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), IETF, Request for Comments: 6020, Category: Standards, ISSN: 2070-1721.
7. Thomas D. Nadeau and Ken Gray. SDN: Software Defined Networks, August 2013
8. Контроллер для T-SDN [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/en/company/netcracker/blog/276643/
9. What’s in OpenDaylight? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mirantis.com/blog/whats-opendaylight/
10. SDN: изменения концепции за 5 лет [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/315524/
11. SDN: кому и зачем это надо? [Електронний ресурс] – Режим доступу до ресурсу: https://www.osp.ru/lan/2012/12/13033012/
12. Software-Defined Networking (SDN) Definition [Електронний ресурс] – Режим доступу до ресурсу: https://www.opennetworking.org/sdn-definition/?nab=1

13. Software-Defined Networking [Електронний ресурс] – Режим доступу до ресурсу: https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN
14. SDN controller [Електронний ресурс] – Режим доступу до ресурсу: https://searchnetworking.techtarget.com/definition/SDN-controller-software-defined-networking-controller
15. What is SDN? [Електронний ресурс] – Режим доступу до ресурсу: https://www.juniper.net/us/en/solutions/sdn/what-is-sdn/
16. What is SDN? [Електронний ресурс] – Режим доступу до ресурсу: https://www.opendaylight.org/what-we-do/what-is-sdn
17. SDN – 10 лет от идеи до реализаций [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/safedata/blog/313320/
18. Эволюция сети к SDN & NFV [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/307356/
19. SDN: новые возможности управления потоками в mesh — сетях [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/239649/
20. Переход от обычной сети ЦОД к SDN [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/huawei/blog/337918/
21. SDN: альтернатива или дополнение к традиционным сетям? [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/hpe/blog/255363/
22. The basics of SDN and the OpenFlow Network Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://noviflow.com/the-basics-of-sdn-and-the-openflow-network-architecture/?gclid=Cj0KCQjw08XeBRC0ARIsAP_gaQBdEwY4jSPG28rPxwhQAZGJ9XnDuGLLQD0q6TCGycQ_JrcJAJp67fcaAkQ-EALw_wcB

23. SDN-архитектура: как извлечь из сетей максимум [Електронний ресурс] – Режим доступу до ресурсу: https://www.comnews.ru/content/110058/2017-10-17/sdn-arhitektura-kak-izvlech-iz-setey-maksimum
24. Software Defined Networking (SDN) - Architecture and role of OpenFlow [Електронний ресурс] – Режим доступу до ресурсу: https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/