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
Завідувач кафедри

_____ Сергій КРАВЧУК
(підпис) (Ім'я, прізвище)

"___"_____2020_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 використання ресурсу хмарних сервісів для побудови мереж SDN (Software-Defined Networking)» _____

Виконав: студент 2 курсу, групи ТЗ - 91мп

(шифр групи)

_____ Шихов Максим Сергійович _____ _____

(прізвище, ім'я, по батькові) (підпис)

Науковий керівник д.т.н., проф. каф. телекомунікацій Романов О.І. _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____

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

Рецензент _____ _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

(підпис)

Київ – 2020 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)

« 20 » січня 2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
_____Шихов Максим Сергійович_____
(прізвище, ім'я, по батькові)

1. Тема дисертації «Дослідження методу використання ресурсу хмарних сервісів для побудови мереж SDN (Software-Defined Networking)»

науковий керівник дисертації д.т.н., проф. каф. телекомунікацій Романов О.І.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» « листопада » 2020 р. № 3208-с

2. Строк подання студентом дисертації 14.12.2020

3. Об'єкт дослідження процес побудови хмарного сервісу з застосуванням надійної програмно-конфігурованої мережі.

4. Предмет дослідження хмарне сховище на основі концепції програмно-конфігурованих мереж.

5. Перелік завдань, які потрібно розробити
1) Виконати огляд літератури з теми дослідження. Розглянути поняття програмно-конфігурованих мереж та хмарних сервісів, дослідити основні сфери застосування;
2) Розглянути методи застосування технологій SDN в побудові хмар;

3) Виконати розробку структурної схеми SDN мережі та розглянути питання надійності.

6. Орієнтовний перелік ілюстративного матеріалу структурна схема програмно-конфігурованої мережі; архітектура хмарного сховища; результати оцінки надійності; результати аналізу стартап-проекту.

7. Орієнтовний перелік публікацій
1) Шихов М.С., Романов О.І. «Побудова контролера мережі SDN з використання хмарних ресурсів». XIV Міжнародна науково-технічна конференція "Перспективи телекомунікацій" ПТ-2020.

8. Консультанти розділів дисертації

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

9. Дата видачі завдання 28.10.2019

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Підбір літератури з теми дослідження	01.11.2019 - 09.06.2020	виконано
2	Підготовка тезисів (Міжнародна науково-технічна конференція «ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ»)	02.04.2020 - 16.04.2020	виконано
3	Написання розділу: Аналіз технологій побудови хмарних сервісів на основі програмно-конфігурованих мереж	10.06.2020 – 20.07.2020	виконано
4	Написання розділу: Аналіз підходів оцінки надійності програмно-конфігурованих мереж	21.07.2020 – 25.08.2020	виконано
5	Написання розділу: метод побудови хмарного сервісу на основі програмно-конфігурованих мереж	26.08.2020 – 17.10.2020	виконано
6	Розробка стартап-проекту	18.10.2020 – 02.12.2020	виконано
7	Оформлення магістерської дисертації	03.12.2020 – 14.12.2020	виконано

Студент _____ Шихов М.С.
(підпис) (ініціали, прізвище)

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

РЕФЕРАТ

Робота містить 86 сторінок, 30 рисунків, 15 таблиць. Було використано 41 джерело.

Мета роботи: розробка методу побудови хмарного сервісу для використання в якості сховища із застосуванням програмно-конфігурованої мережі та дослідження надійності отриманої системи.

Об'єкт дослідження: процес побудови хмарного сервісу з застосуванням надійної програмно-конфігурованої мережі.

Предмет дослідження: хмарне сховище на основі концепції програмно-конфігурованих мереж.

В представленій роботи розглянуто концепцію програмно-конфігурованих мереж, наведено основні сфери їх застосування та представлений розвиток технології. Показана актуальність застосування SDN для побудови великих за топологією мереж, наведені особливості застосування програмно-конфігурованих мереж в побудові хмарних сервісів. Розроблено структурну схему програмно-конфігурованої мережі для побудови хмарного сховища та досліджено питання надійності та відмовостійкості розробленого рішення.

Ключові слова: програмно-конфігуровані мережі, хмарні технології, хмарне сховище, надійність, відмовостійкість.

ABSTRACT

The work contains 86 pages, 30 figures, 15 tables. 41 sources were used.

Purpose: to develop a method of building a cloud service for use as a repository using a software-configured network and to study the reliability of the resulting system.

Object of research: the process of building a cloud service using a reliable software-configured network.

Subject of research: cloud storage based on the concept of software-configured networks.

The presented work considers the concept of software-defined networks, presents the main areas of their application and presents the development of technology. The relevance of the application of SDN for the construction of large topology networks is shown, the features of the application of software-defined networks in the construction of cloud services are given. The structural scheme of the program-defined network for construction of cloud storage is developed and questions of reliability and fault tolerance of the developed decision are investigated.

Keywords: software-defined networking, cloud technologies, cloud storage, reliability, fault tolerance.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	Application Programming Interface – інтерфейс програмування застосунків
BYOD	Bring your own device – IT-політика, згідно з якою співробітникам дозволено або рекомендується використовувати особисті мобільні пристрої (телефони, планшети, ноутбуки) для доступу до корпоративних даних та систем
IBN	Intent-Based Networking – модель мереж на основі намірів
NBI	Northbound Interface
ONF	Open Networks Foundation – Фонд відкритих мереж
SaaS	System as a Service – програма як послуга
SBI	Southbound Interface
SDN	Software-Defined Networking – програмно-конфігуровані мережі
ВМ	Віртуальна машина
ОС	Операційна система
ПЗ	Програмне забезпечення
ЦОД	Центр обробки даних

ВСТУП

Сучасні тенденції, такі як зростання числа підключених до Інтернету пристроїв, експоненціальне зростання обсягів інформації, розвиток хмарних технологій, BYOD, Big Data, на очах змінюють корпоративний телеком. Йде нарощування обсягів мережевого трафіку, і у бізнесі все частіше виникає необхідність конфігурувати великомасштабні мережі. Спростити цю задачу можуть технології програмно-конфігурованих мереж SDN, які дозволяють перевести мережеві елементи під контроль програмного забезпечення, зробити їх більш інтелектуальними, полегшити управління ними.

SDN може допомогти, тому що мета управління мережею - дозволяти різним пристроям (неважливо - що належить чи компанії, співробітникам, а також різних виробників) підключатися до мереж і використовувати їх ресурси з обмеженнями, заснованими на принципах «хтось що-де-як-чому» при кожному підключенні. Це вимагає постійних застосувань політик серед всіх пристроїв. Надалі, адміністратор, який змінює політики, нічого очікувати змушений проводити години, роблячи зміни в кожному пристрій окремо, і ці зміни повинні узгоджуватися по всьому підприємству. Ось в чому роль SDN. Вони надають узгоджене, відносно швидке управління мережами, дозволяючи зміни у всій мережі з єдиною консолі управління.

Також важливим є те, що механізм віртуалізації мереж побудований на базі вільного програмного забезпечення, що дозволяє мережевим адміністраторам швидше і ефективніше управляти великими потоками даних з однієї консолі. За даними Infonetics Research у 2019 році обсяг SDN досягнув 13 млрд. доларів. В порівнянні з 781 млн. доларів у 2014 році.

Хмарні сервіси (public cloud services) - це програми і платформи, які розміщуються і працюють на серверах провайдерів хмарних послуг. Принцип роботи хмарних платформ досить простий. Хмарне сховище надається користувачеві в необхідному обсязі, оплачується за фактом використання і позбавляє від необхідності купувати власну IT-інфраструктуру для зберігання

даних і керувати нею. Це забезпечує гнучкість, швидку масштабованість і надійність.

Роль програмно визначених мереж у хмарних обчисленнях дозволяє користувачам швидко реагувати на зміни. Управління SDN робить конфігурацію мережі більш ефективною та покращує продуктивність та моніторинг мережі.

Програмно-визначена мережа, дозволяє безпосередньо програмувати мережеве управління для програм та мережевих служб. Реалізована за допомогою програмного забезпечення мережева архітектура відокремлює функції управління мережею та переадресацію від фізичного обладнання, наприклад маршрутизаторів та комутаторів, для створення більш керованої та динамічної мережевої інфраструктури.

1. АНАЛІЗ ТЕХНОЛОГІЙ ПОБУДОВИ ХМАРНИХ СЕРВІСІВ НА ОСНОВІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

1.1.Програмно-конфігуровані мережі

1.1.1. Основні поняття

Програмно-конфігуровані мережі (SDN) - це відносно нова парадигма для проектування, побудуви та експлуатації мереж. Вона є продовженням дослідницької роботи, що велася в Університеті Каліфорнії в Берклі та Стенфордському університеті в 2008 році, і вона все більше набирає обертів як з дослідницької, так і з точки зору промисловості у секторі комп’ютерних мереж.

Рушійною мотивацією була необхідність серйозного зрушення в технологіях мереж для забезпечення набагато простішої конфігурації, управління, функціонування, змін в конфігурації та еволюції, ніж у сучасних комп’ютерних мережах. Дійсно, експлуатація та управління мережею є складними завданнями, та оператори телекомунікацій, а також провайдери Інтернету та хмарних послуг стикаються з великими проблемами при налаштуванні великих мереж, застосуванні бажаних політик та впровадженні нових технологій [1]. На сьогоднішній день комп'ютерні мережі лежать в основі більшості критичних інфраструктур та багатьох служб, з якими пов’язана наша повсякденна діяльність - будь то ділова, споживча, соціальна чи приватна. Конфігурація та управління великими мережами дуже складна, оскільки дотримання політик високого рівня вимагає вказувати їх з точки зору команд низького рівня багатьох власних вертикально інтегрованих пристроїв різних постачальників [1]. Ці труднощі заважають розробці та швидкому забезпеченню нових вдосконалених протоколів та послуг для найвибагливіших сучасних та майбутніх стаціонарних та мобільних додатків. Така ситуація призвела до визначення нової парадигми, що передбачає багатошарову архітектуру мережі.

Ключовою концепцією (рис. 2.1) є розділення логіки управління мережею від мережевого обладнання, яке спрямовує та транспортує трафік [2]. Традиційні мережі орієнтовані на апаратне забезпечення та більшість мережевих пристроїв (наприклад, маршрутизатори та комутатори) закриті в тому сенсі, що вони включають обидва частини управління та даних (рис. 2.1a) і мають інтерфейси від власних постачальників. Заміна або просто оновлення протоколів та служб є дуже складним, оскільки все обладнання потрібно замінити / оновити [3].

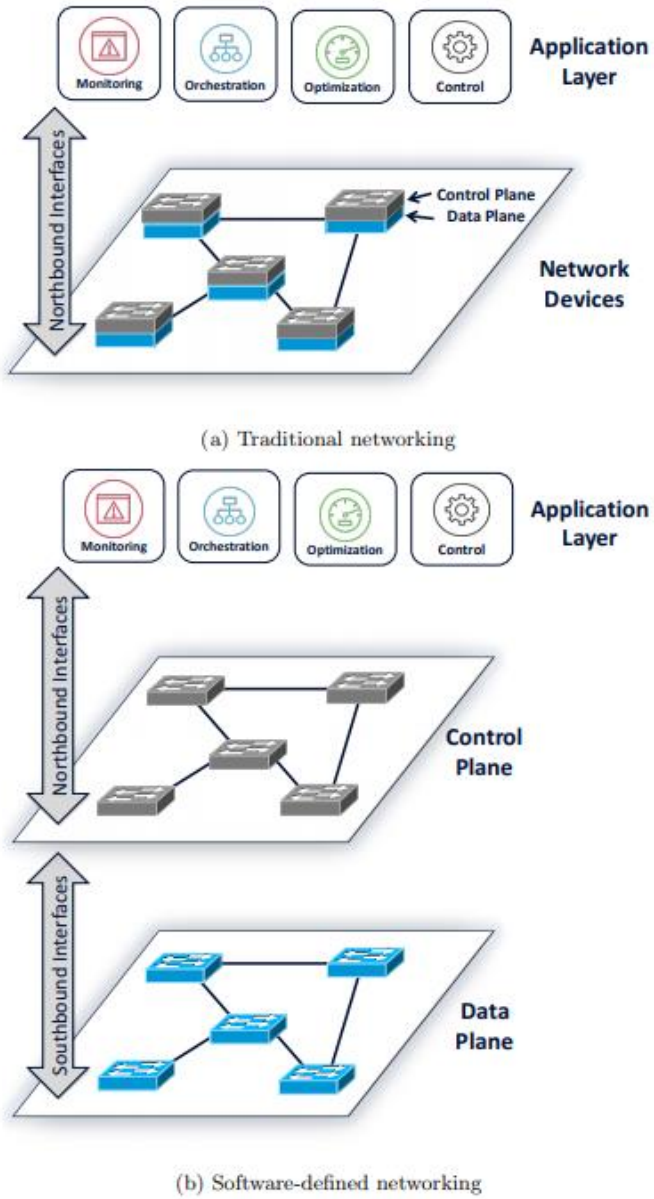


Рис 1.1 Поділ даних та контролю у SDN.

У SDN (Рис 2.1b), мережеві пристрої - це просто пристрої переадресації пакетів, які постійно перебувають у так званій площині даних, тоді як “мозок” (програмована управляюча логіка) знаходиться в площині управління, чітко і вище площини даних; мережеве обладнання програмується через рівень управління.

1.1.2. Архітектура SDN

Як показано на рис. 2.2, поділ рівнів здійснюється за допомогою відкритих інтерфейсів програмування (API), що називаються північний інтерфейс або Northbound Interface (NBI, щодо програм) і південний інтерфейс або Southbound Interface (SBI, між площиною управління та даними). Ця концепція дозволяє співіснувати новій парадигмі з традиційною; справді, декілька реалізацій сучасного комерційно доступного мережевого обладнання є гібридними, що підтримують як новий SBI, так і традиційні протоколи. Це повинно полегшити перехід до архітектури SDN.

Логічна сутність, що розміщує програмне забезпечення, визначає основні мережеві послуги в площина управління (наприклад, маршрутизація, аутентифікація, виявлення) відома як SDN-контролер (або просто контролер). Він відповідає за виконання політик та послуг, що вимагаються програмами, видаючи команди та отримуючи події та інформацію про стан від пристроїв у площині даних (іменованих як Комутатори з підтримкою SDN) через SBI.

У науково-технічній літературі SDN було докладено багато зусиль для вивчення Southbound Interface. Основним API на SBI є OpenFlow [4], стандартизований Фондом відкритих мереж (див. Розділ 2.1.3) і підтримується багатьма провідними постачальниками мережевого обладнання (наприклад IBM, NetGear, NEC, HP).

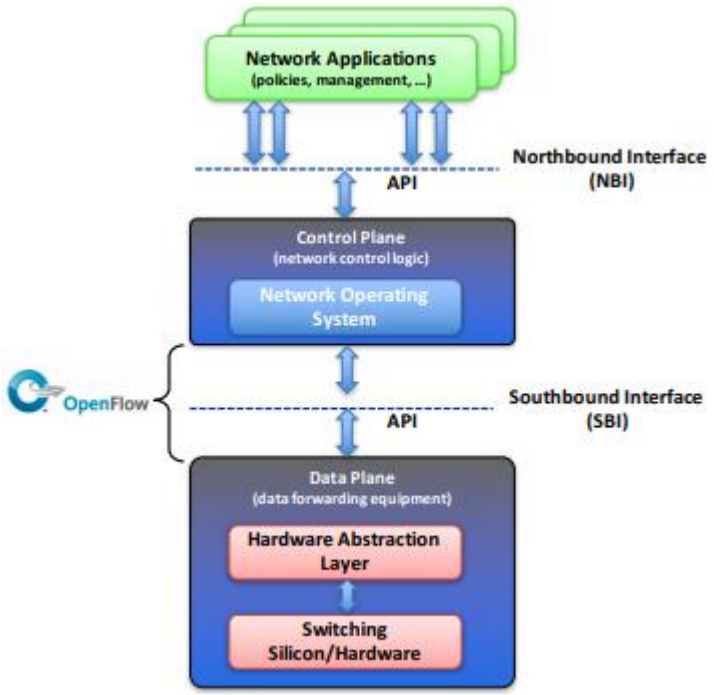


Рис 1.2 Архітектура та інтерфейси SDN

До цього часу менше акценту було зроблено на північному інтерфейсі та протоколи. NBI відповідає за надання засобів для уточнення та запиту мережеві політики та послуги абстрактним способом, незалежно від того, як вони приводяться в дію контролером. Перспективна пропозиція щодо північного інтерфейсу - це модель мереж на основі намірів (Intent-Based Networking, IBN) [9] [10] прийнята у проекті ONOS;

В останні роки як з технологічної, так і з промислової точки зору розпочато кілька ініціатив для сприяння розвитку SDN. Головною ініціативою є Фонд відкритих мереж (ONF) - неприбуткова організація створена у 2011 році такими компаніями, як Deutsche Telekom, Facebook, Google, Microsoft, Verizon та Yahoo!. В даний час ONF налічує десятки компаній-партнерів і її місія - “просування та впровадження програмно-конфігурованих мереж через розробку відкритих стандартів”. Ще однією відповідною некомерційною ініціативою є Відкрита мережева лабораторія (Open Networking Lab, ON.Lab), створена постачальниками послуг, мережевими операторами та постачальниками мережевого обладнання з основною метою побудувати відкритих інструментів та платформ SDN. Наприкінці 2016 року ONF та

ON.Lab оголосили , що вони приєднаються в 2017 році до ONF, щоб пришвидшити прийняття SDN.

Першим досягненням ONF є стандарт OpenFlow™, що дозволяє дистанційне програмування площини переадресації [5]. OpenFlow забезпечує інтерфейс між площинами управління та даних, що забезпечує безперебійний зв'язок між компонентами на двох рівнях. Спочатку OpenFlow був запропонований для експериментів із технологіями та застосуваннями в мережі кампусів [4]. Потім він набрав обертів, аж до його визначення як стандарту ONF для південного інтерфейсу між елементом управління та площиною даних.

Проект ONOS створений для розробки відкритого бачення мереж наступного покоління на основі SDN, щоб вийти за межі поточних мереж, які є «закритими, власними, складними, дорогими в використанні, негнучкими». Явною метою є «створення мережевої операційної системи з відкритим кодом, яка дозволить постачальникам послуг створювати справжні програмно-конфігуровані мережі». Цю мету переслідує спільнота партнерів, у тому числі багато найбільших промислових гравців у цій галузі, будь то оператори мережі, інтернет-провайдери, постачальники хмарних служб та центрів обробки даних, а також постачальники серед яких AT&T, Cisco, Ericsson, Google, Huawei, NOKIA, NEC, NTT Communications, Samsung, Verizon.

Проект сприяв розробці ONOS™ (Операційна система відкритих мереж), яка, як стверджується, є першою мережевою операційною системою SDN з відкритим кодом. ONOS - не перший контролер SDN з відкритим кодом, все ж це перша цільова масштабованість, висока доступність та висока продуктивність. Він задуманий для подолання обмежень попередніх контролерів такі як NOX [7] та Beacon [8], які були тісно пов'язані з OpenFlow API та надавали додатки з прямим доступом до повідомлень OpenFlow - у цьому сенсі вони не забезпечили належного рівня абстракції SDN для додатків, звідси необхідність справжньої операційної системи мережі SDN. Зверніть увагу, що термін мережева операційна система (NOS) згадувався

кілька десятиліть тому та позначав операційні системи з мережевими функціями (наприклад, від Novell); це застаріле використання було змінено на [7] “для позначення систем, які забезпечують середовище виконання для програмного управління мережею ”. Таке визначення наразі все ще мається на увазі коли даний термін вживається в контексті SDN, що, мабуть, і є причиною того, що сьогодні поняття SDN-контролер та мережева операційна система часто використовуються як взаємозамінні.

З точки зору ONOS модель мереж, що базується на намірах (IBN), відіграє важливу роль у визначенні потреб мережі через службу управління політиками. Ідея IBN полягає в тому, що програми повинні надсилати запити або правила до площини управління у вигляді намірів, тобто що треба зробити, а не як треба зробити (гасло звучить так: «скажи мені, що тобі потрібно, а не як це зробити!"). Простим прикладом наміру є запит на встановлення взаємозв'язку "точка-точка" між двома вузлами, доповнений виконанням або вимоги до послуг, такі як мінімальна пропускна здатність та тривалість.

Намір можна розглядати як об'єкт, що містить запит до мережевої операційної системи для зміни поведінки мережі. Він може складатися з:

- Мережевих ресурсів, частини мережі, на які впливає намір;
- Обмеження, такі як пропускна здатність, оптична частота та тип лінії зв'язку;
- Критерій, що описує частину трафіку, на яку впливає намір;
- Інструкції, тобто дії, що застосовуються до зрізу трафіку, що нас цікавить.

Модель IBN абстрагує специфікації потреб робочих навантажень, що споживають мережеві послуги ("що"), від того, як мережева інфраструктура задовольняє ці потреби ("як"). В теорії це досягається через службу управління політиками на північному інтерфейсі SDN-контролера; остання відповідає за

переклад мережевої політики у відповідні контрольні дії, наприклад встановлення правил потоку.

Наміри визначають на логічному рівні запитувані дії, а потім - SDN відповідає за їх задоволення шляхом безпосередньої взаємодії з фізичними пристроями. Роблячи це, структура IBN абстрагує складність мережі дозволяючи мережевим операторам та програмам описувати політики, а не інструкції низького рівня для пристроїв.

SDN полегшує віртуалізацію мережі і, отже, спрощує реалізацію таких функцій, як динамічна реконфігурація мережі (наприклад, у багатонаціональних середовищах). Однак важливо визнати, що основні можливості технологій SDN безпосередньо не забезпечують цих переваг. Деякі функції SDN та їх основний внесок у покращення віртуалізації мережі:

• Розділення площини управління та площини даних: розділення між планами управління та площинами даних в архітектурах SDN, а також стандартизація інтерфейсів для зв'язку між цими рівнями дозволило концептуально уніфікувати різні мережеві пристрої постачальників під однаковими механізмами управління. Для цілей віртуалізації мережі абстракція, що надається площиною управління та розділенням площини даних, полегшує розгортання, налаштування та оновлення пристроїв у віртуалізованих мережевих інфраструктурах. Розділення площини управління також вводить ідею мережевих операційних систем, яка складається з масштабованої та програмованої платформи для управління та організації віртуалізованих мереж.

• Мережева програмованість: Програмованість мережевих пристроїв є одним з основних внесків SDN у віртуалізацію мережі. До появи SDN віртуалізація мережі була обмежена статичною реалізацією накладених технологій (таких як VLAN), завдання, делеговане адміністраторам мережі та логічно розподілене серед фізичної інфраструктури. Можливості програмування, представлені SDN, забезпечують динаміку, необхідну для

швидкого масштабування, обслуговування та налаштування нових віртуальних мереж. Більше того, програмованість мережі також дозволяє створювати власні мережеві додатки, орієнтовані на інноваційні рішення віртуалізації мережі.

– Логічно централізоване управління: абстракція пристроїв площини даних, що надаються архітектурою SDN, дає мережевій операційній системі, також відомій як система оркестрування SDN, єдиний вигляд мережі. Таким чином, це дозволяє програмам користувацького управління отримувати доступ до всієї топології мережі з логічно централізованої платформи управління, що дозволяє централізувати конфігурації та керувати політикою. Таким чином, розгортання та управління мережевими технологіями віртуалізації стає простішим, ніж на початку розподілених підходів.

– Автоматизоване управління: архітектура SDN покращує платформи віртуалізації мережі, забезпечуючи підтримку автоматизації адміністративних завдань. Централізоване управління та можливості програмування, надані SDN, дозволяють розробляти власні мережеві додатки для створення та управління віртуальною мережею. Автомасштабування, управління трафіком та QoS - це приклади засобів автоматизації, які можна застосувати до віртуальних мережевих середовищ.

Серед різноманіття сценаріїв, коли SDN може вдосконалити реалізацію мережевої віртуалізації, можна згадати тестові зразки мережі кампуса, корпоративні мережі, багатонаціональні центри обробки даних та хмарних мереж. Незважаючи на таке успішне застосування технологій SDN у таких сценаріях використання віртуалізації мережі, багато роботи є

необхідні як для вдосконалення існуючої мережевої інфраструктури, так і для вивчення потенціалу SDN для вирішення проблем віртуалізації мережі. Приклади включають додатки SDN для таких сценаріїв, як домашні мережі,

корпоративні мережі, пункти обміну Інтернетом, стільникові мережі, мережі радіодоступу Wi-Fi та спільне управління програмами кінцевого хосту.

1.1.3. Надійність SDN

Надійність системи чи послуги - це концепція, що охоплює кілька атрибутів якості (рис. 2.3), а саме доступність, безпеку, конфіденційність, цілісність та ремонтопридатність, при цьому конфіденційність, цілісність та доступність є частиною складеного захисту атрибутів.

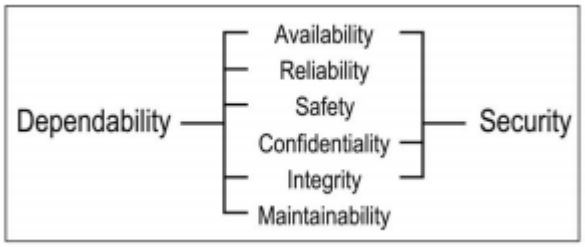


Рис 1.3: Атрибути надійності

Надійність виражає безперервність правильного обслуговування. Це ймовірність того, що система функціонує належним чином в інтервалі часу *(0, t):*

$$R(t) = P(!відмова\ на\ (0,\ t)) \qquad (1.1)$$

Як правило, надійність оцінюється за допомогою широко розповсюджених показників, таких як Середній час до відмови ($C_{ЧДВ}$), Середній час між відмовами ($C_{ЧМВ}$) та Середній час до ремонту ($C_{ЧДР}$).

Доступність виражає готовність до правильного обслуговування. Це ймовірність того, що система функціонує належним чином у момент часу t:

$$A(t) = P(!відмова\ на\ t)) \qquad (1.2)$$

і це часто виражається як час безвідмовної роботи, поділений на загальний час (час безвідмовної роботи плюс час простою); часто це обчислюється як відношення:

$$A = \frac{C_{ЧДВ}}{C_{ЧДВ} + C_{ЧДР}} = \frac{1}{1 + \frac{C_{ЧДР}}{C_{ЧДВ}}} \qquad (1.3)$$

що показує, що для покращення доступності важливо зменшити співвідношення між $C_{ЧДР}$ та $C_{ЧДВ}$, збільшуючи середній час до відмови та / або скорочуючи середній час на ремонт.

Безпека - це відсутність катастрофічних наслідків для користувачів та навколишнього середовища. Конфіденційність - це властивість захисту інформації; це властивість системи мати можливість не надавати доступ, не розголошувати та не надавати зрозумілу (захищену) інформацію неавторизованим особам, організаціям чи процесам. Цілісність - це властивість, що відображає відсутність неналежних змін, якщо вони стосуються, наприклад, системи, повідомлень або даних. Ремонтопридатність - це здатність системи зазнавати змін та ремонтів.

У програмно-конфігурованій мережі, контролер є фізично розподіленою сутністю хоча і логічно централізованою. Це пояснюється тим, що вимоги до надійності - головним чином щодо масштабованості, доступності та надійності - вимагають її проектування в розподіленій архітектурі. Якщо, як стверджувалось, SDN стане технологією майбутніх мереж, він повинен повністю їх вирішити вимоги. Наприклад, оператори телекомунікацій навряд чи приймуть SDN для заміни існуючих мереж рівня оператора, якщо не буде доведено, що SDN може забезпечити принаймні таку ж якість обслуговування, забезпечуючи при цьому більшу гнучкість та легкість управління.

Масштабованість не є строго атрибутом надійності в поточній класифікації, проте вона є основною проблемою для SDN. Хоча це, безумовно, є фундаментальним аспектом розробки для контролерів SDN, це питання часто ігнорується. Поширена думка, що - на відміну від поточних мереж, де пристрої часто реалізуються за допомогою спеціалізованих інтегральних мікросхем (ASIC), - логічно централізований, але фізично розподілений контролер, визначений програмним забезпеченням, може не масштабуватися в міру зростання мережі. Однак, схоже, що це помилкове припущення. Не існує не штучного обмеження для масштабованості SDN. Проблеми масштабованості в SDN подібні як у будь-якій розподіленій системі, і

масштабованості за своєю суттю важче досягти, ніж у традиційних мережах. Тобто, якщо розподілений SDN необхідний для забезпечення єдиного загально мережевого уявлення, рішення повинні включати протоколи розподіленої узгодженості. Якщо існують обмеження для розподілених систем, це, мабуть, ті, що зазначені у відомій теоремі Брюера, стверджуючи, що не завжди можливо досягти коректності, високої доступності та стійкості до збоїв вузлів.

Що стосується доступності, то вимоги щодо SDN дуже жорсткі; вони не відрізняються від аналогічних для сучасних мереж. Якщо SDN доводиться використовувати для базових, але критично важливих послуг, таких як телефонія, вони повинні надати наскрізну доступність п'яти "дев'яток" (тобто 99,999%), як у сучасних мережах операторського рівня. П'ять "дев'яток" становлять близько п'яти хвилин простою на рік. У сучасних мережах це досягається за рахунок ручної конфігурації та тривалого часу розгортання. Гнучкість SDN є дуже привабливою з точки зору операторів телекомунікаційного зв'язку , але вони не збираються жертвувати доступністю для ремонтопридатності. Досягти таких високих рівнів доступності, коли мережа визначена програмно, може бути важко, і це вимагає ретельного проектування та точного впровадження, але це можливо. Проблеми доступності систем SDN сягають глибше, ніж ті, що випливають із їхньої складності.

Запобігання несправностям означає запобігання виникненню або виникненню несправностей.

Відмовостійкість означає уникнення несправностей обслуговування при наявності несправностей.

Усунення несправностей означає зменшення кількості та кількості несправностей.

Прогнозування несправностей означає оцінку теперішньої кількості, майбутньої здатності до несправностей та ймовірних наслідків несправностей. Запобігання несправностям і стійкість до несправностей мають на меті

забезпечити можливість надання послуги, якій можна довіряти, тоді як усунення несправностей та прогнозування несправностей мають на меті досягнення впевненості в цій здатності, обґрунтовуючи, що технічні характеристики та надійність та безпека є адекватними та що система ймовірно, зустріне їх.

Життєвий цикл системи складається з двох фаз: розробки та використання. Етап розробки включає всі дії, починаючи з презентації початкової концепції користувача і закінчуючи рішенням про те, що система пройшла всі приймальні випробування та готова надавати послуги в середовищі свого користувача. На етапі розробки система взаємодіє із середовищем розробки, і дефекти розробки можуть вноситися в систему середовищем. Середовище розробки системи складається з наступних елементів:

1) фізичний світ з його природними явищами;
2) люди-розробники, дехто, можливо, не має компетенції або має зловмисні цілі;
3) засоби розробки: програмне та апаратне забезпечення, що використовується розробниками, щоб допомогти їм у процесі розробки;
4) виробничо-випробувальні приміщення.

Етап використання системи починається з того часу, коли система приймається до використання та починає надання своїх послуг користувачам. Використання складається з чергування періодів правильного надання послуги (що називатиметься наданням послуг), відключення послуги та відключення послуги. Збої в роботі сервісу спричинені збоєм у роботі послуги. Це період, коли через інтерфейс сервісу доставляється неправильна послуга (включаючи взагалі жодну). Відключення послуги - це навмисне припинення обслуговування уповноваженим суб’єктом. Дії з технічного обслуговування можуть виконуватися протягом усіх трьох періодів фази використання. На етапі використання система взаємодіє із середовищем використання та може

негативно впливати на несправності, що виникають у ній. Середовище використання складається з таких елементів:

1) фізичний світ з його природними явищами;
2) адміністратори (включаючи супровідників): організації (люди чи інші системи), які мають повноваження управляти, модифікувати, ремонтувати та використовувати систему; деякі уповноважені люди можуть не мати компетентності або мати зловмисні цілі;
3) користувачі: сутності (люди або інші системи), які отримують послуги від системи на їх інтерфейсах використання;
4) провайдери: організації (люди або інші системи), які надають послуги системі на інтерфейсах її використання;
5) інфраструктура: організації, які надають системі спеціалізовані послуги, такі як джерела інформації (наприклад, час, GPS тощо), лінії зв'язку, джерела живлення, потік охолоджуючого повітря тощо.
6) зловмисники: зловмисні організації (люди та інші системи), які намагаються перевищити будь-які повноваження, які вони можуть мати, або змінити службу, або зупинити її, змінити функціональність або продуктивність системи або отримати доступ до конфіденційної інформації. Прикладами можуть бути хакери, вандали, корумповані інсайдери, агенти ворожих урядів чи організацій та шкідливе програмне забезпечення.

Використовуваний тут термін технічне обслуговування, що застосовується після загального використання, включає не тільки ремонт, але й усі модифікації системи, які мають місце на етапі використання системи. Отже, технічне обслуговування - це процес розробки, і попереднє обговорення розробки стосується і технічного обслуговування. Різні форми обслуговування узагальнені на рис. 3.

Примітно, що технічне обслуговування та відмовостійкість - це пов’язані поняття; різниця між відмовостійкістю та технічним обслуговуванням у цьому документі полягає в тому, що технічне обслуговування передбачає участь зовнішнього агента, наприклад, ремонтника, випробувального обладнання, дистанційного перезавантаження програмного забезпечення. Крім того, ремонт є частиною усунення несправностей (на етапі використання), і прогнозування несправностей зазвичай враховує ситуації ремонту. Насправді ремонт можна розглядати як відмовостійкість у більшій системі, що включає систему, що ремонтується, та людей та інші системи, які виконують такий ремонт.

Відмовостійкість є (поряд із запобіганням, усуненням та прогнозуванням несправностей) засобом підвищення надійності системи. Це, безумовно, має першочергове значення для SDN, оскільки у разі виходу з ладу контролера може бути скомпрометована вся мережа, оскільки від неї залежать усі програми та служби. Всі контролери SDN розроблені з механізмами, що дозволяють переносити подібні події, і явно стійкість до несправностей є одним з основних методів, що використовуються для забезпечення високого рівня стійкості. У цій роботі пропонується введення відмов як метод тестування для навмисного впровадження відмов, що представляють події відмов, які насправді можуть відбуватися на різних рівнях у мережах SDN, з метою оцінки механізмів відмовостійкості контролера та, загалом, для оцінки ступеня, в якому контролер здатний забезпечити бажаний рівень стійкості.

Поняття стійкості (або відмовостійкості) має кілька визначень, оскільки воно розвивалося в різних дисциплінах, включаючи фізику, психологію, екологію, інженерію. Швидше за все, цей термін спочатку відносився до властивості фізичного матеріалу або сутності, але зараз ця концепція застосовується також до мережевих систем або організацій. Незважаючи на різні визначення, цитуючи, у більшості з них є “три елементи: здатність змінюватись, коли діє сила, діяти адекватно або мінімально, поки сила діє,

повернутися до заздалегідь визначений очікуваний нормальний стан, коли сили ослаблюють або роблять неефективними ”.

В техніці цей термін якось інтуїтивно передає поняття здатність протистояти ненавмисним або зловмисним загрозам (збоям, атакам тощо) та відновлювати нормальні умови роботи; ми можемо сказати, що стійкість системи часто розглядається як її здатність забезпечувати та підтримувати прийнятний рівень продуктивності чи обслуговування в разі відмов. Варто прямо вказати, що стійкість - це властивість системи в макроскопічному масштабі, тобто властивість системи в цілому.

Стійкість комп’ютерної мережі - це здатність забезпечити і підтримувати прийнятний рівень обслуговування в умовах несправностей та проблем із нормальною роботою. Стійкість мережі визначається як здатність мережі захищатись та підтримувати прийнятний рівень сервісу за наявності таких проблем, як шкідливі атаки, несправності програмного та апаратного забезпечення, людські помилки (наприклад, неправильне налаштування програмного та апаратного забезпечення) та масштабних стихійних лих, що загрожують його нормальній роботі.

Стійкість систем оцінюється за допомогою залежного від часу показника, відомого як показник якості $F(\bullet)$, який є кількісним показником ефективності роботи системи. Як показано на малюнку 2.4, стан системи характеризується значенням $F(\bullet)$, на який безпосередньо впливають дві події (руйнівна подія та відповідна дія відновлення).

Можна визначити декілька показників, щоб забезпечити вимірювання стійкості щодо надійності, шляхів підключення до мережі, потоків тощо.

На рисунку 1.4 система спочатку є “функціональною” в момент часу t_0, і цей стан залишається постійним до настання руйнуючої події в момент t_e, приводячи значення функції доставки системи від початкового значення $F(t_0)$ до нижчого значення $F(t_d)$. Таким чином, передбачається, що система

функціонує в погіршеному режимі від t_e до t_d, коли досягає точки, коли функціональність вважається повністю втраченою.

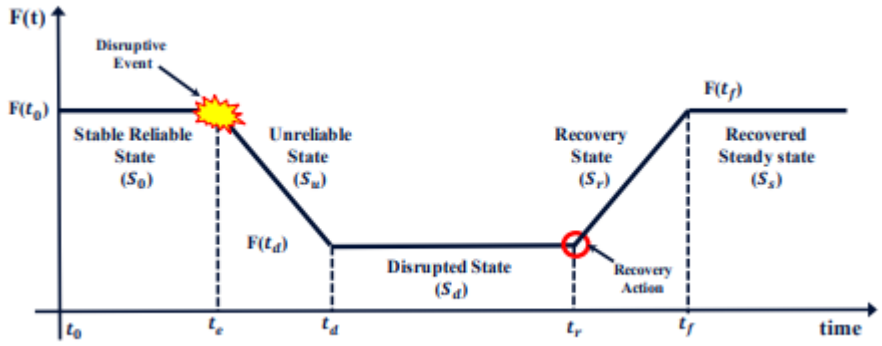


Рис 1.4: Поведінка системи у випадку руйнуючої події.

Система залишається в такому стані до тих пір, поки не почнеться дія відновлення / ремонту в момент t_r, коли система відновить свою функціональність, хоча і в погіршеному режимі. В результаті дії стійкості система вважається відновленою і повністю функціональною в момент часу t_f, із значенням функції доставки $F(t_f)$. Однак остаточний стан, досягнутий системою після дії відновлення, не обов'язково повинен збігатися з початковим станом системи, тобто показником $F(t_f)$ може бути рівним, більшим або меншим за $F(t_0)$. Нарешті, значення стійкості, що відповідає конкретній функції показника заслуги, можна обчислити як:

$$\Lambda\ (t) = F(t) - F(t_d)F(t_0) - F(t_d) \qquad (1.4)$$

де $0 \leq \Lambda(t) \geq 1$ для $t \in (t_r, t_f)$, припускаючи, що дії відновлення вдається відновити функціональність. В останні роки концепція стійкості була розширена, щоб включити поняття надійності також стосовно змін; справді, це все частіше сприймається як здатність системи залишатися надійною в сучасних змінах. Це випливає з роботи Лапрі, який вказував на необхідність вирішити питання зростання складності сучасних настільки поширених обчислювальних систем, що випливає із змін, які можуть бути функціональними, екологічними та технологічними.

Лапрі представив масштабовану стійкість як концепцію "живучості при безпосередній підтримці нової поширеності обчислювальних систем".

Для цілей цієї роботи буде використано наступне визначення: стійкість мережі - це здатність забезпечувати та підтримувати прийнятний рівень обслуговування в умовах відмов.

У наш час стійкість є основною вимогою та метою проектування комп’ютерні мережі. Це пов’язано з тим, що комп’ютерні мережі лежать в основі більшості критичних інфраструктур, що піддаються як ненавмисним помилкам / збоям, так і зловмисним (кібер-)атакам.

Незважаючи на велику літературу про SDN, існує кілька відкритих проблем, пов'язаних із виконанням вимог SDN до надійності та стійкості. Однією з причин цього є те, що майже немає практичного способу експериментувати з новими протоколами в достатньо реалістичних умовах (наприклад, у масштабі, що несе реальний трафік), щоб отримати впевненість, необхідну для їх широкого розгортання.

Доступність може створювати більше загроз для SDN, ніж ті, що створюються загальні розподілені системи. Проблеми доступності систем SDN мають більш глибоке коріння, ніж ті, що випливають із складної та критичної взаємозалежності між різними протоколами мережі та розподілених систем, які вони використовують. Автори наводять для цього аргумент, який варто представити тут.

Розглянемо мережу на рис. 2.5, де C1 - C5 - це репліки контролера в розподіленій площині управління, а S1 - S9 - перемикачі в площині даних.

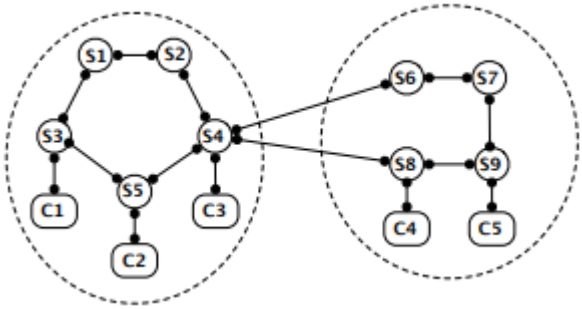


Рис 2.5: Випадок загрози доступності в SDN

Також зазначимо, що розподілені контролери включають консенсусні протоколи, такі як Paxos, щоб забезпечити їх послідовне уявлення про топологію мережі та стан площини даних навіть за наявності збоїв або відключень. Коли зв'язки між S4 і S6 і S4 і S8 виходять з ладу, виникає мережевий розділ. У цьому випадку перемикачі в розділі площини даних праворуч не можуть бути оновлені, оскільки вони можуть контактувати лише група меншості (C4 і C5) копій контролерів. Якщо шлях між S6 і S8, керований площиною управління до розділення, не повністю міститься в правому розділі, S6 і S8 не можуть зв'язуватися, навіть якщо між ними є шлях, на який не впливає несправність. Такі критичні ситуації підривають високу доступність. Очевидно, що застарілі мережеві протоколи не страждають від цієї проблеми: коли відбуваються розділи, маршрутизатори знову сходяться до нових маршрутів внутрішнього розділу. Цитуючи авторів, випадок, який вони наводять показує, що "поточні конструкції SDN не забезпечують важливих властивостей відмовостійкості, що робить SDN менш доступними, ніж традиційні мережі в деяких ситуаціях". Хоча існують рішення описаної проблеми (наприклад, використання розділеного консенсусу), вони можуть вирішити її лише частково; зокрема, у випадку супутніх подій можуть бути підводні камені. Автори самі запропонували рішення - хоча й не довели - на основі

Однак проблеми доступності в SDN є більш глибокими, ніж це, і виникають навіть за наявності ідеальних каналів управління, які не накладають ніяких залежностей.

Розглянемо мережу, зображену на малюнку 1. Коли зв’язки S4 S6 і S4 S8 виходять з ладу і мережа розділена, площина даних розділу праворуч не може бути оновлена, оскільки комутатори там можуть контактувати лише з меншістю групи реплікаційної площини управління (тобто C4 і C5). Найважливіше, що S6 і S8 не можуть взаємодіяти між собою, якщо раніше налаштований шлях між ними не повністю міститься в новому розділі. У §3 ми стверджуємо, що прості виправлення лише цієї адреси вирішують проблему доступності. Важливо те, що застарілі розподілені протоколи, такі як OSPF, не страждають від цієї проблеми: коли відбуваються розділи, маршрутизатори OSPF знову сходяться до нових маршрутів внутрішнього розділу. Іншими словами, поточні проекти SDN не надають важливого значення властивості відмовостійкості, що робить SDNsless доступними, ніж традиційні мережі в деяких ситуаціях!

У цьому розділі ми описуємо конструкцію відмовостійкої тканини SDN, яка може забезпечити високий ступінь доступності при наявності різного роду відмов у мережі. Наш дизайн представляє переосмислення архітектури SDN, від каналу управління до площини управління, для вирішення залежностей та уникнення підводних каменів, що є суттєвими для SDN сьогодні. Ми представляємо дизайн як низку уточнень, починаючи з базового дизайну, при цьому кожне вдосконалення стосується певного аспекту доступності або узгодженості контролера.

Ми починаємо з базового дизайну, який гарантує послідовність стан контролера і дозволяє комутаторам / контролерам надійно спілкуватися, але не обов'язково забезпечує високу доступність.

Він включає два набори механізмів.

Надійна повінь: Щоб повністю звільнити канал управління від усіх залежностей, ми підтримуємо канал управління, заснований на надійному затопленні. Такий канал управління може бути встановлений незалежно від маршрутизації або транспортних протоколів.

Реплікаційні контролери: Контролери зроблені стійкими до відмов, використовуючи реплікацію автоматів та надійне затоплення.

Зокрема, контролери використовують протокол консенсусу, такий як Paxos [14], щоб отримати консенсус від кворуму (як правило, це проста більшість контролерів) перед тим, як реагувати на зміни в топології та конфігурації системи. Наприклад, щоразу, коли відбувається зміна топології (наприклад, зниження зв'язку або подія відновлення посилання), контролери приходять до консенсусу, що відповідна подія буде наступною подією, яка буде оброблятися контролерами і передавати комутаторам новий набір правил пересилання на основі оновленої топології.

Усі зв’язки між контролерами для досягнення консенсусу та зв’язки від контролерів до комутаторів обробляються за допомогою надійного затоплення. Цей базовий дизайн надійний до несправностей контролерів (поки більшість з них живі), а також може надійно обробляти зв'язок між контролерами та комутаторами, навіть якщо правила переадресації точка-точка застарілі щодо поточної топології.

Підводні камені: Незважаючи на те, що базовий дизайн забезпечує стійкість до певних типів відмов (наприклад, мережа працездатна, навіть якщо деякі контролери виходять з ладу або є збої в деяких шляхах між контролерами та комутаторами), він не є надійним до відмов цей розділ встановлений контролером. Розглянемо, наприклад, мережу, зображену на малюнку 1. Якщо посилання S4 S6 та S4 S8 не працюють, то мережа розділена. У цьому випадку політики маршрутизації та мережі для розділу праворуч не можуть бути оновлені. Це особливо виснажує ситуацію, якщо існуючі маршрути, що з'єднують пари комутаторів у правій частині розділу, не повністю містяться всередині розділу. Наприклад, якщо маршрут з S6 до S8 - це S6! S4! S8, коли мережа стає розділеною, тоді маршрут не може бути оновлений до робочого шляху S6! S7! S9! S8, оскільки комутатори S6 і S8 можуть контактувати лише з двома з п'яти контролерів (тобто C4 і C5) і, отже, не зможуть скласти кворум більшості для обробки оновлення топології. Навіть якщо комутатори

налаштовані на використання швидкого відмови у формі заздалегідь налаштованого шляху резервного копіювання, зв’язок між S6 - S8 не гарантується, оскільки шлях резервного копіювання також може бути не повністю розміщений всередині розділу.

Таким чином, базовий дизайн забезпечує послідовну обробку динамічних змін топології мережі, але не забезпечує високої доступності у випадку мережевих розділів. Ретроспективно очевидно, що базовий дизайн успадковує сильні та слабкі сторони, пов'язані з надійними розподіленими послугами; це лише прямий наслідок програмно визначеної мережі та її розділення площини управління у логічно централізовану службу контролера. Можливість виконувати операції управління мережею залежить від того, чи може служба контролера збирати кворуми для обробки мережевих подій. Обмеження доступності, пов'язані з розподіленими послугами (наприклад, теорема CAP [3]), можна подолати лише за допомогою певної форми підтримки з боку базової мережі, що зосереджується у решті цього розділу.

Проста модифікація вищезазначеної конструкції полягає у забезпеченні комбінації алгоритму знімків Ченді-Лампорта, надійного затоплення та консенсусу цілого кворуму. Однак такі механізми є дорогими, і хоча вони видаються достатніми, незрозуміло, чи всі вони необхідні.

Шарма та ін. зосередити увагу на відмовостійкості та відновленні несправностей у OpenFlow для його розгортання в мережах операторського класу, як засіб покращення стійкості SDN. Мережі операторського класу вимагають суворої вимоги, що мережа повинна відновлюватися після поломки протягом інтервалу 50 мс. Вони показують, що OpenFlow може бути не в змозі задовольнити цю вимогу, і вони пропонують дію відновлення в комутаторах без залучення контролера. Що стосується надійності, вони коротко обговорюють в якості майбутньої роботи деякі можливі підходи.

Виклики стійкості на майбутнє мережі на базі SDN грунтуються на аналізі відключень поточних мереж рівня оператора, щоб визначити три

основні фактори, що впливають на ефективність їх механізмів відмови, а саме: неперевірений операційний контекст, численні збої під час вікна відмови та людські / процедурні помилки. Виходячи з цього, отримані наслідки включають:

- Потреба в нових методах стійкості та в нових виробничих підходах до тестування. Це пояснюється тим, що SDN, як очікується, зменшить час розгортання служби до порядку секунд; це, в свою чергу, може означати зменшення тестування та оцінки.
- Потреба в нових механізмах відмови. Це пояснюється тим, що багато механізмів запрограмовано в сучасних мережевих пристроях, тоді як у SDN відмовостійкий процес вимагатиме взаємодії між різними рівнями та між різними доменами.
- Бізнес-моделі послуг, керовані Угодами про рівень обслуговування (SLA), вимагатимуть розширених специфікацій стійкості та перевірки. Це пов’язано з тим, що SDN надає високу гнучкість, надання послуг буде більш динамічним (наприклад, через міграцію для оптимізації ресурсів), а рівень обслуговування потрібно буде динамічно перевіряти.

B4 – практичний приклад широкомасштабної реалізації глобальної мережі глобальної мережі на основі SDN, що з'єднує центри обробки даних Google с по всьому світу. Автори надають деталі реалізації одного з перших і найбільших розгортань SDN, який продемонстрував свою ефективність у задоволенні вимог як до продуктивності, так і до надійності. Однак, незважаючи на видатні показники, B4 зазнав відключення через людську помилку. Дійсно, під час планової роботи з технічного обслуговування два фізичні комутатори були налаштовані з однаковим ідентифікатором, що спричиняло значні помилки відхилення зв'язку, тобто їх мережеві інтерфейси постійно йшли вгору і вниз, і більше заходів з обробки протоколів для виявлення топології мережі. Це призвело до подальших збоїв у

загальнодоступній мережі Google, що вплинуло на мережеве підключення їхніх клієнтів.

Непередбачувані події підкреслюють, що до тих пір, поки система виявляється надійною в результаті випробувальної діяльності, тим не менше вона може вийти з ладу у виробничому середовищі, де умови експлуатації змінюються та розвиваються з часом. Більше того, традиційні засоби тестування програмного забезпечення є недостатніми для перевірки стійкості таких складних розподілених систем проти всіх можливих сценаріїв відмов, які можуть охоплювати весь стек системи.

1.2.Хмарні сервіси

1.2.1. Основні поняття

Існувало багато визначень хмарних обчислень різних дослідників. Barkley RAD визначає хмарні обчислення як: «Хмарні обчислення стосуються як програм, що надаються як послуги через Інтернет, так і апаратного та системного програмного забезпечення в центрах обробки даних, які надають ці послуги. Самі послуги вже давно називають Програмним забезпеченням як послугою (SaaS). Апаратне та програмне забезпечення центру обробки даних - це те, що ми будемо називати хмарою. Коли хмара стає доступною для широкого загалу в платному режимі, ми називаємо її публічною хмарою; послуга, що продається, - Utility Computing. Ми використовуємо термін Private Cloud для позначення внутрішніх центрів обробки даних бізнесу чи іншої організації, недоступних для широкої громадськості. Таким чином, хмарні обчислення - це сума SaaS та комунальних обчислень, але не включає приватні хмари. Люди можуть бути користувачами або постачальниками SaaS, або користувачами, або постачальниками службових обчислень ".

Короткий опис особливостей хмарних обчислень, описаних Станоевською-Слабєвою та Возняком, є:

• Хмарні обчислення - це нова обчислювальна парадигма.

• Інфраструктурні ресурси (апаратне забезпечення, сховище та системне програмне забезпечення) та додатки надаються в режимі X-як-сервіс. Коли ці послуги пропонуються незалежним постачальником або зовнішнім клієнтам, хмарні обчислення базуються на бізнес-моделях використання платника.

• Основними особливостями хмар є віртуалізація та динамічна масштабованість на вимогу.

• Обчислювальні програми та SaaS надаються інтегровано, навіть якщо обчислювальні програми можуть використовуватися окремо.

• Хмарні послуги використовуються або через веб-браузер, або через визначений API.

Як я вже згадував, що хмарні обчислення - це еволюція від Grid Computing, тому можна сказати, що більшість підприємств перейшли з Grid на Cloud. Отже, зараз я буду вивчати, як це впливає на підприємства після переходу з Grid на Cloud. Іншими словами, я побачу, що мали Grid Computing і чим володіють Cloud Computing зараз, щоб допомогти економіці підприємства.

У “Хмарній міграції: приклад міграції IT-системи підприємства на IaaS”, Хадже-Хоссейні та ін. (2010a) розповів про сторонню хмарну інфраструктуру. За їхніми словами, якщо буде введена хмарна інфраструктура третьої сторони, то це представляє багато можливостей для підприємств покращити управління доходами та витратами як для фінансового персоналу, так і для клієнтів. Це також сприяє полегшенню управління грошовим потоком для фінансових ресурсів, оскільки хмарна модель ціноутворення має мінімальні авансові витрати та щомісячні рахунки, а також зменшує мінливість витрат на електроенергію. Це переваги порівняно із внутрішнім центром обробки даних, оскільки придбання обладнання може коштувати дорого, а грошовий потік може бути повільним та складним для клієнтів. Поряд з цим витрати на енергію також зменшаться, оскільки ви не керуєте своїм власним центром обробки даних, і за це буде відповідати стороння хмара. Хмарна

інфраструктура також дуже корисна для фінансового відділу компанії для зменшення адміністративного навантаження. Сторонні хмарні інфраструктурні рішення пропонують нові моделі ціноутворення, які допомагають управляти доходами для клієнтів, продажів та маркетингового персоналу.

Хадже-Хосейні та ін. (2010a) дійшов висновку, що хмарні обчислення - це руйнівна технологія, яка має змінити спосіб розгортання IT-систем на підприємствах через її дешевий, простий і масштабований характер. Хмарні обчислення можуть бути значно дешевшими порівняно із придбанням та обслуговуванням власного центру обробки даних, оскільки усувають проблеми, пов’язані з підтримкою, оскільки немає фізичної інфраструктури для обслуговування. Однак існує багато соціально-технічних питань, які підприємства повинні розглянути перед тим, як перейти на Cloud.

На будь-якому підприємстві адміністративні витрати на низькому рівні можуть бути досить високими, оскільки відділи розкидані по будівлі, часто набагато більші, ніж витрати на обладнання. За допомогою хмари підприємства можуть розвантажити три види адміністрації низького рівня. По-перше, це системна інфраструктура, яка включає технічне обслуговування обладнання, запасні частини, додавання нових машин та програмне забезпечення інфраструктури, про що піклується хмара. По-друге, як тільки підприємства визначають політику резервного копіювання, хмарний постачальник відповідає за її виконання. Нарешті, одна програма встановлюється один раз і стає доступною для всіх авторизованих користувачів. Хоча управління додатком, тобто підтримка додатків, проблеми оновлення та управління користувачами, не включаються, оскільки перехід до хмари не сильно змінює ці завдання. Важливо зазначити, що витрати на низькому рівні іноді можуть перевищувати загальні витрати на хмарну послугу.

У звичайних системах використання системних ресурсів є низьким, оцінюється в 15–20% для центрів обробки даних; інші оцінки нижчі. Існує

багато причин низького використання, оскільки менеджери, як правило, купують для майже пікових та майбутніх навантажень, а отже, не використовують весь потенціал весь час. Хоча, щоб допомогти в цьому питанні, хмарні обчислення згладжують ці ефекти для багатьох клієнтів, і сьогодні можуть досягти 40% використання.

Потужність сервера є дорогою через такі процеси, як охолодження та інші витрати електроенергії. Якщо їх об'єднати разом, вони можуть дорівнювати вартості одного типового сервера, який використовується сьогодні. Хмарні провайдери можуть зробити набагато краще, ніж типові серверні центри, завдяки кращому керуванню перетвореннями напруги, прохолодному клімату та кращому охолодженню, а також нижчим тарифам на електроенергію (постачальники хмарних послуг, як правило, скупчуються біля гідроенергетики). Хмарні провайдери також зазвичай розташовані там, де нерухомість недорога.

Розенталь та інші у своїй статті "Хмарні обчислення: нова бізнес-парадигма для обміну біомедичною інформацією", обговорювали три основні фактори витрат біомедичних підприємств та те, як на них впливає технологія хмарних обчислень. Вони включають системне адміністрування, непрацюючу потужність, а також споживання енергії та засоби.

Маюр та інші досліджує службу зберігання даних Amazon S3 для науково-дослідних додатків. На їх думку, це пакети S3 за єдиним методом ціноутворення для всіх трьох характеристик даних, тобто високої довговічності, високої доступності та швидкого доступу, але більшість програм не потребують усіх об'єднаних. Наприклад, архівне зберігання; який потребує довговічності, але може вижити при меншій доступності та продуктивності доступу.

Отже, пропонується, що S3 повинен надавати послуги за допомогою ряду обмежених класів послуг, щоб користувачі могли вибрати бажану сумісність довговічності / доступності / доступу до кращих витрат. Отже,

вартість вища, якщо тривалість / доступність / продуктивність групи послуг зберігання зберігають разом.

У звіті McKinsey & Co «Прочищення повітря про хмарні обчислення» вони заявляють, що хмарні обчислення можуть коштувати вдвічі дорожче, ніж власні центри обробки даних. Однак це питання лише для великих підприємств, але малі та середні підприємства це не зачіпає, і вони отримують вигідні витрати. На їх думку, "хмарні пропозиції в даний час є найбільш привабливими для малих та середніх підприємств ... і більшість клієнтів хмар - це малий бізнес" (Люблінський та Борис, 2009). Причиною цього є те, що менші компанії не мають можливості перетворитися на гігантські центри обробки даних. Вартість мінливість є ключовим аспектом хмарних обчислень, і коли підприємства обирають прозорість витрат, масштабованість та мінливість витрат, виникає новий виклик та можливість.

1.2.2. Архітектура типового хмарного сервісу

NIST (Національний інститут стандартів та технологій) є добре прийнятою установою у всьому світі для їх роботи у галузі інформаційних технологій. Я представлю робоче визначення, надане NIST хмарних обчислень. NIST визначає архітектуру хмарних обчислень, описуючи п’ять основних характеристик, три моделі хмарних служб і чотири моделі розгортання хмари.

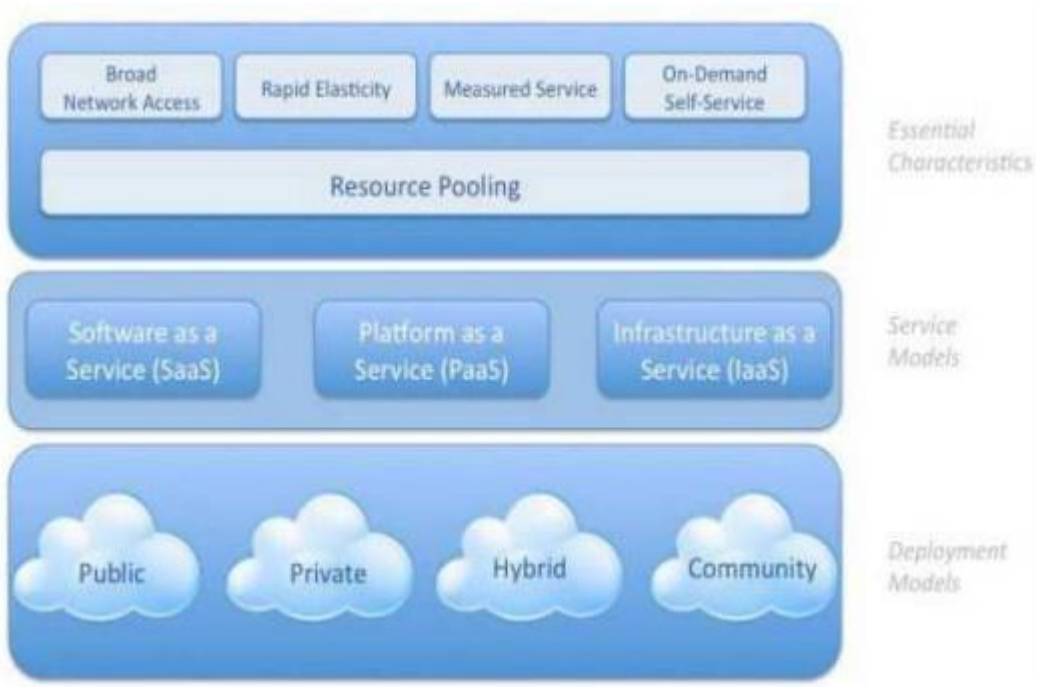


Рис 1.1 - Візуальна модель робочого визначення NIST хмарних обчислень

Основні характеристики хмарних обчислень

Як описано вище, існує 5 основних характеристик хмарних обчислень, що пояснює їх взаємозв'язок та відмінність від традиційних обчислень.

• Самообслуговування на замовлення

Споживач може надавати або припиняти надання послуг, коли це необхідно, без взаємодії людини з постачальником послуг.

• Широкий доступ до мережі

Він має можливості через мережу та доступ до нього через стандартний механізм.

• Об’єднання ресурсів

Обчислювальні ресурси провайдера об'єднані для обслуговування кількох споживачів, які використовують багатокористувацьку модель, з різними фізичними та віртуальними ресурсами, що динамічно призначаються, залежно від попиту споживачів.

• Швидка еластичність

Послуги можуть бути надані швидко та еластично.

• Помірне обслуговування

Системи хмарних обчислень автоматично контролюють і оптимізують використання ресурсів, надаючи можливість вимірювання типу послуг (наприклад, зберігання, обробка, пропускна здатність або активні облікові записи користувачів).

Моделі хмарних сервісів

Існує 3 моделі хмарних служб, і ці 3 основні класифікації часто називають «моделлю SPI», тобто програмним забезпеченням, платформою чи інфраструктурою як послугою.

• Хмарне програмне забезпечення як послуга

Це можливість, за допомогою якої споживач може використовувати програми постачальника, що працюють у хмарі.

• Хмарна платформа як послуга

У цьому типі послуги споживач може розгорнути, споживач створив або придбав додатки, створені за допомогою мов програмування або інструментів, наданих постачальником, на хмарній інфраструктурі.

• Хмарна інфраструктура як послуга

Це можливість, що надається споживачеві, завдяки якій він може надавати обробку, зберігання, мережі та інші основні обчислювальні ресурси, де споживачі можуть розгортати та запускати програмне забезпечення (тобто операційні системи, додатки).

Моделі хмарного розгортання

• Громадська хмара

Хмарна інфраструктура доступна широкому загалу.

• Приватна хмара

Тип хмари, який доступний виключно для однієї організації.

• Хмара спільноти

У цьому типі хмарної моделі розгортання, інфраструктура хмари є спільною для кількох організацій і підтримує певне співтовариство із спільними проблемами.

• Гібридна хмара

Це хмарна інфраструктура, яка складається з двох або більше хмар, тобто приватної, спільноти або загальнодоступної.

Переваги масштабу

Фактом є те, що всі типи заходів безпеки, що застосовуються в більших масштабах, дешевші. Таким чином, прийнявши на роботу хмарні обчислення, підприємства отримують кращий захист за однакової суми грошей. Безпека включає всі види захисних заходів, таких як фільтрація, управління виправленнями, зміцнення екземплярів віртуальних машин, управління людськими ресурсами та їх контроль, перевірка апаратного та програмного забезпечення, надійна автентифікація, ефективний контроль доступу на основі ролей та рішення щодо об'єднаного управління ідентифікацією за замовчуванням. , що також покращує мережеві ефекти співпраці між різними партнерами, що беруть участь у обороні. Поряд з цими перевагами, іншими перевагами є:

Кілька місць:

Хмарні провайдери за замовчуванням мають економічні ресурси для тиражування вмісту, і це збільшує надмірність та незалежність від невдач. Отже, це забезпечує аварійне відновлення.

Крайові мережі:

Хмарні обчислення забезпечують надійність, підвищення якості та менше проблем з локальною мережею для підприємств завдяки розміщенню сховища, обробки та доставки ближче до краю мережі.

Покращені терміни реагування (інциденти):

Хмарні провайдери мають більше інцидентів або добре запущені масштабніші системи. Ці системи допомагають покращити терміни реагування, наприклад через раннє виявлення нових розгортань шкідливого програмного забезпечення, воно може розробити більш дієві та ефективні реакції на інциденти.

Управління загрозами:

Малі підприємства не мають ресурсів для найму спеціалістів для вирішення конкретних проблем безпеки, але хмарні провайдери можуть це зробити та забезпечити краще управління загрозами.

1.2.3. Вартість хмарних сервісів

Економічна привабливість хмарних обчислень часто згадується як «перетворення капітальних витрат на операційні витрати». Підприємства, які використовують хмарні обчислення, платять по-різному, залежно від угоди між ними та провайдерами хмарних обчислень. Зазвичай постачальники хмарних обчислень мають деталізовані моделі калькуляції витрат, які використовуються для виставлення рахунків користувачам на основі оплати за користування.

На ринку хмарних обчислень доступні різні моделі вартості. Однак найбільш використовувана модель обговорюється Armbrust, яка є короткостроковою моделлю виставлення рахунків.

Armbrust описує короткострокову модель виставлення рахунків як одну з найцікавіших та нових особливостей хмарних обчислень.

Дослідники обговорили економіку хмарних обчислень у двох аспектах, тобто споживча перспектива та перспектива постачальника. Обидві перспективи мають різні моделі собівартості / ціни.

З точки зору споживачів я обговорюю моделі витрат, які приймаються постачальниками для оплати споживачами. Отже, у цьому поданні ми бачимо моделі ціноутворення з погляду споживача.

Згідно з Armbrust, хмарні обчислення надають модель калькуляції, тобто платять за використання обчислювальних ресурсів на короткостроковій основі, коли це потрібно, а також звільняють їх, коли не потрібно. Отже, таким чином ви відпускаєте машини та сховища, коли вони вже не є корисними (Armbrust et al., 2009, p3).

Наприклад, Elastic Compute Cloud (EC2) від Amazon Web Services (AWS) продає 1,0-ГГц x86 ISA «зрізи» за 10 центів на годину, і якщо ви хочете додати новий «зріз» або екземпляр, його можна додати за 2 до 5 хвилин. Служба масштабованого зберігання даних Amazon (S3) стягує від 0,12 до 0,15 доларів США за гігабайт на місяць, а якщо ви хочете отримати додаткову пропускну здатність, то для переміщення даних з AWS через Інтернет потрібно 0,10–0,15 доларів США за гігабайт. Отже, Amazon заявляє, що шляхом статистичного мультиплексування кількох екземплярів на одному фізичному ящику цей ящик можна орендувати багатьом клієнтам, які не будуть заважати один одному.

Armbrust називає цей метод калькуляції витратами як "платіть як хочете". Наприклад, якщо ви купуєте години роботи у хмарних обчисленнях, вони можуть бути розподілені нерівномірно за часом у мережевому співтоваристві, тобто сьогодні використовують 200 серверних годин, а завтра не годину на сервері, і платити лише за те, що ви використовуєте. Хоча така оплата може бути дорожчою, ніж придбання подібного сервера за той самий період, але Armbrust стверджує, що ціна перевищує переваги еластичності та перенесення ризиків від хмарних обчислень. Що стосується еластичності

хмарних обчислень, то можливість додавати або видаляти ресурси з дрібним зерном (по одному серверу за один раз), а також використаний час хвилин, а не годин або тижнів, дозволяє більш точно відповідати ресурсам для навантаження.

Використання сервера в реальному світі оцінюється від 5% до 20%. Це здається досить низьким, але це зауваження, що середнє робоче навантаження для багатьох служб перевищує в 2–10 разів. Деякі користувачі навмисно вказують пік, менший за очікуваний, оскільки вони повинні вказати пік, але натомість вони дозволяють ресурсам простоювати в не пікові часи. Це призводить до марної витрати ресурсів.

На ринку є й інші моделі, що стосуються споживачів. Вони прийняли одну з трьох форм, тобто багаторівневе ціноутворення, ціноутворення за одиницю та ціноутворення на основі передплати. Amazon хмара прийняла багаторівневу модель ціноутворення, при якій хмарні послуги пропонуються в кілька рівнів, і кожен рівень забезпечує фіксовані обчислювальні характеристики (тобто виділення пам'яті, тип і швидкість процесора тощо) та SLA (Угода про рівень обслуговування) за певною ціною за одиницю часу. Ціни на перуніт в основному використовуються при передачі даних та використанні пам'яті. Програма GoGrid Cloud використовує розподіл основної пам’яті, де вони позначають “ОЗУ / годину” як одиницю використання для своєї системи (GoGrid, 2010). Цей метод є більш гнучким, ніж багаторівневе ціноутворення, оскільки дозволяє користувачам перерозподілити місце пам’яті на основі своїх потреб. Нарешті, модель на основі підписки в основному використовується для SaaS. Ця модель дозволяє користувачам прогнозувати періодичні витрати на використання хмарних обчислень. Деякі дослідники працювали з витратами на хмарні центри обробки даних. Грінберг та ін. описав, як можна зменшити витрати на хмарний центр обробки даних, маючи на увазі вартість серверів, інфраструктури, потужності та мереж. Відповідно до них, витрати можна зменшити, працюючи з центрами обробки даних при більш холодних температурах, щоб зменшити витрати на

охолодження та будуючи центри мікро-даних для зменшення вартості пропускної здатності.

Як я вже згадував, що хмарні обчислення - це еволюція від Grid Computing, тому можна сказати, що більшість підприємств перейшли з Grid на Cloud. Отже, зараз я буду вивчати, як це впливає на підприємства після переходу з Grid на Cloud. Іншими словами, я побачу, що мали Grid Computing та чим володіють Cloud Computing зараз, щоб допомогти економіці підприємства.

Для підприємств, крім інвестування витрат на хмарні обчислення, важливо знати і вартість надання послуг хмарних обчислень з кількох причин.

По-перше, існує ймовірність того, що підприємства не можуть легально мігрувати на державні хмари, отже використання приватних хмар стає все більш важливим. По-друге, якщо підприємства одного разу запустили приватну хмару, вони завжди можуть орендувати її запасний IT-простір. Тому з цієї причини підприємствам добре знати, скільки коштує приватна хмара.

1.2.4. Застосування SDN для розгортання хмарного сервісу

SDN відкрив закритий та розподілений характер управління мережевими переадресаційними елементами та мережами загалом. Це надає абсолютно нову можливість розглядати мережі як програмно керований апаратний ресурс. Це основний інструмент для багатьох нових та інноваційних мережевих додатків та пропонує новий підхід та розуміння віртуалізації мережі. Наслідки дуже широко розповсюджені. Вони досягають простих віртуальних локальних мереж (VLAN), подібних механізму поділу або інкапсуляції, до повної віртуалізації мережевих ресурсів або навіть мережі загалом, що добре відомо з хмарних обчислень.

FlowVisor

Інструмент FlowVisor став першою розробкою в цьому новому напрямку. Це було логічним наслідком попиту чи питання, як

використовувати різні контролери одночасно на одній і тій же підкладці. Тож питання полягало в тому, як поділитися фізично доступними ресурсами, наприклад тестувати різні мережеві програми без більших перешкод. Народилася ідея інструменту під назвою FlowVisor. FlowVisor - це лише проксі-сервер каналу управління OpenFlow з додатковим адміністративним інтерфейсом для визначення відповідних політик користувача.

Користувач отримує мережевий фрагмент, який підключений до його контролера. FlowVisor обробляє лише команди OpenFlow згідно з раніше визначеними політиками зрізів та забезпечує їх застосування. Детальніше, він пересилає контрольні повідомлення OpenFlow, які не порушують політики зрізів. Він скидає повідомлення та надсилає повідомлення про помилку для команд, які роблять це.

Це дає можливість використовувати кілька контролерів та додатків в одній фізичній мережі. Наприклад, це дає можливість протестувати нові алгоритми безпосередньо у продуктивній мережі, не впливаючи на це.

OpenVirteX

Насправді FlowVisor був дуже успішним, але, на жаль, основний дизайн та його принципи працюють лише для стандарту протоколу 1.0. Коли кілька версій таблиць були представлені з OF версії 1.1, було зрозуміло, що потрібно знайти інше рішення. Знову ж таки була реалізована інноваційна ідея - повна віртуалізація мережі.

OpenVirtex використовує механізм відображення внутрішнього протоколу Інтернету (IP), який зіставляє кожен трас, що входить до мережі, керованої OpenVirteX, на елементі переадресації вхідних пакетів до адреси приватного мережевого управління класу IP. Це використовується для відокремлення трас, що належить різним користувачам / контролерам. Він повністю прозорий, оскільки користувач отримує повноцінну віртуалізовану мережу. Він навіть не може зробити жодних висновків щодо базової фізичної

топології, подібно до того, як повноцінний віртуалізований хост не знає про базове обладнання.

Наразі OpenVirtex доступний в альфа-версії. На жаль, останні дослідження щодо використання цього інструменту в дослідницьких середовищах, як у розділі 3, описаному тестовому стенді OFELIA TUB, свідчать про те, що ще не весь можливий набір функцій повністю реалізований і працює. І все-таки ця реалізація є першим справжнім мережевим гіпервізором на основі SDN.

1.3.Висновки до розділу

Підсумовуючи, нові підходи необхідні для автоматизованої перевірки стійкості SDN під час реалізації реальних умов навантаження, а саме на виробництві. Інноваційні підходи дозволять виявляти точки відмов, які в іншому випадку важко виявити за допомогою просто тестування програмного забезпечення, допомагаючи розробити кращі методи виявлення та створивши правильні засоби пом'якшення для відновлення системи при виникненні реальних проблем.

У двох словах можна сказати, що вартість та безпека є важливими факторами для будь-якого підприємства для адаптації хмарних обчислень.

Підприємства, як правило, вибирають способи оплати для постачальників послуг хмарних обчислень.

Найпоширеніший спосіб оплати для підприємств та хмарних провайдерів - це "платіть по ходу".

Це одна з нових функцій хмарних обчислень, яка в довгостроковій перспективі дешевша, ніж наявність власного центру обробки даних. Однак хмарні обчислення дешевші для малих та середніх підприємств, ніж великі. Еластичність - ще один фактор для підприємств, що адаптують хмарні обчислення, оскільки вони можуть динамічно використовувати свої ресурси. Amazon S3 надає послуги в комплекті з високою довговічністю, високою

доступністю та швидким доступом, однак більшості підприємств потрібні не всі, крім деяких. Отже, хмарні провайдери повинні працювати над цим, щоб надати підприємствам більше переваг. Адміністративні витрати зменшуються із прийняттям Cloud Computing для підприємств.

Безпека даних є найсерйознішим питанням підприємств, що адаптують хмарні обчислення.

Більшість питань безпеки пов’язані з відсутністю контролю підприємства над фізичною інфраструктурою.

Веб-служби та веб-браузери також викликають занепокоєння щодо хмарних обчислень, оскільки більшість хмарних служб доступні через Інтернет. Обов'язки постачальників та споживачів щодо безпеки залежать від хмарних моделей, наприклад Amazon AWS EC2, пропонований IaaS, несе відповідальність постачальників за управління фізичною, екологічною та віртуалізаційною безпекою. З іншого боку, споживач відповідає за безпеку на рівні IT-системи, тобто операційну систему, програми та дані. Чим нижче стек зупиняється хмарним постачальником, тим більше безпеки споживач відповідає за впровадження та управління.

Окрім архітектури, управління та операційні домени також важливі для безпеки даних хмари. Питання управління включають управління та управління ризиками підприємств, юридичне та електронне виявлення, дотримання та аудит, управління життєвим циклом інформації, а також портативність та сумісність. У той час як операційні питання включають традиційну безпеку, безперервність бізнесу та відновлення після катастроф, роботу центру обробки даних, реагування на аварії, сповіщення та виправлення, захист програм, шифрування та управління ключами, а також віртуалізацію.

Поряд із проблемами безпеки, є також деякі переваги хмарних обчислень щодо безпеки. Ці переваги включають переваги масштабу з кількома місцями, крайові мережі, імпровізований часовий ряд реагування та

управління загрозами. Інші переваги включають безпеку як ринковий диференціатор, стандартні інтерфейси для керованих служб безпеки, швидке та розумне масштабування ресурсів, аудит та збір доказів, краще управління ризиками, концентрацію ресурсів та ефективні оновлення та за замовчуванням. Подібно до цього, Болдінг також згадав про сім переваг хмарних обчислень, включаючи централізовані дані, реагування на інциденти, тестування захисту паролів, реєстрацію, покращений стан програмного забезпечення для захисту, збірки та тестування безпеки.

2. АНАЛІЗ ПІДХОДІВ ПОКРАЩЕННЯ НАДІЙНОСТІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

2.1Загальний огляд проблеми

Оскільки SDN-контролер є єдиним елементом, який може відмовити в процесі експлуатації, слід розглянути різні механізми для забезпечення відмовостійкості мережі. Забезпечення надійності мережі базується на ідентифікації відмови та резервуванні. Крім того, у мережах з новими технологіями відновлення має відбуватися протягом 50 мс або швидше. Сучасна тенденція мережевої взаємодії із завданнями конфігурації програмного забезпечення полягає у переході від фізичного до вищих рівнів, що також підвищує надійність додатків. Цей програмний метод переходить від апаратного методу резервування до програмно-забезпечуваного.

Є багато нових переваг при переході від надмірності на фізичному (апаратному) рівні до надмірності на вищому рівні моделі OSI. По-перше, програмне забезпечення має більшу здатність резервування, ніж апаратне. Крім того, операції резервування на вищих рівнях більш очевидні, ніж на нижчих рівнях моделі. Наприклад, якщо негативний результат введення надмірності реєструється після спроби резервування фізичного рівня (SDH), додаткове резервування доведеться виконати на мережевому рівні, і ця операція повинна розпочатися з деякою розрахунковою затримкою, що збільшує час перемикання і важко реалізується [12].

Тому надмірність у мережах SDN, де надмірність вирішується на рівні програми (на площині управління SDN), та її впровадження демонструється в площині даних SDN, залежність від надмірності на інших рівнях менш важлива, а швидкість комутації зростає.

На цей час існують високі вимоги до надійності (стійкості до відмов) програмно-конфігурованих мереж, а також особливостей відновлення після відмови, крім того, це відновлення повинно враховуватися замовником.

Щоб забезпечити стійкість до несправностей мережі, усі методи можна розділити наступним чином:

- Захисне перемикання (або резервування);

- Відновлення (або перемаршрутизація).

Процес резервування відбувається шляхом перенаправлення трафіку на з'єднання, підготоване до з'єднання запасним шляхом. Відновлення – це пошук нового шляху (перемаршрутизація) після відновлення, спричиненого відмовою.

Кожна установка має свої переваги та недоліки для забезпечення відмовостійкості (Таблиця 2.1).

Таблиця 2.1 Переваги і недоліки методів забезпечення відмовостійкості

Метод	Переваги	Недоліки
Резервування	Швидке поновлення зв’язку	Підвищені вимоги до пропускної здатності
Перемаршрутизація	Краще використовує пропускну здатність мережі	Довге відновлення зв’язку Високий ризик нестабільності мережі

Обидва ці методи дозволяють надавати користувачеві індикацію готовності необхідного з'єднання або індикацію готовності різних послуг, що йому надаються. Вибираючи послугу мережі, доступність послуги є більш важливим показником, важливішим за інші параметри QoS (наприклад, затримка, втрата пакетів тощо). Якщо роздивитись поточний ринок телекомунікаційних послуг, то побачимо, що більше половини користувачів очікують доступності на рівні 99,9%, тож ви повинні надати 0,99999999 $K_г$ ((для бізнес-сектору), що становить близько 50 мс.

Надійність мережевої інфраструктури забезпечується завдяки використанню зв'язку між алгоритмами захисного перемикання та відновлення.

Існує декілька реалізацій технічних рішень для забезпечення надійності SDN при використанні одного або двох SDN- контролерів, а також одного сервера або кластера серверів. Перевага використання декількох серверів полягає в тому, що якщо один сервер вийде з ладу, розриву не буде. Однак використання декількох контролерів SDN є економічно недоцільним для малих мереж (призводить до втрат переваги над традиційними мережами передачі даних).

Враховуючи доступні методи забезпечення надійності, а також їх комплексне використання в різних технічних реалізаціях SDN можна оцінити надійність мережі.

Для досягнення цієї мети необхідно вирішити проблему, а саме:

- Демонстрація побудови мережі за концепцією SDN;
- Розглянути можливі методи забезпечення надійності;
- Оцінити ефект від введення надмірності контролера SDN;
- Запропонувати вирішення проблеми перенаправлення;

Надійність мережевої інфраструктури забезпечується використанням алгоритмів резервування та відновлення зв'язку між вузлами мережі та підвищення надійності самих вузлів, особливо комутаторів. Сьогодні для всіх критичних технічних рішень потрібні модулі управління з можливостями, що ілюструють надмірність різних підсистем.

При проектуванні мережі необхідно намагатися мінімізувати як можливість відмови, так і вплив відмови. Це непросте завдання, оскільки існує взаємодія Взаємозв'язок між зменшенням ймовірності відмови та мінімізацією впливу відмови. Сучасні телекомунікаційні мережі - це мережі, які мають

велику пропускну здатність і зазвичай використовують волоконно-оптичні лінії зв'язку. Тому забезпечення структурної надійності таких мереж є вирішальним.

Надлишковість та відновлення - два основні підходи, що забезпечують структурну надійність телекомунікаційних мереж на ходу

Система вузлів і ліній зв'язку. Основними вимогами до методу надійності є:

- Економія смуги пропускання;
- Обмеження обчислювальних ресурсів;
- Коефіцієнт заміщення;
- Складність запропонованих способів;
- Масштабованість.

Слід звернути увагу на те, що оптимізація будь-яких показників за наявності обмежень у більшості випадків є складним завданням. Для нього можуть бути використані різні методи. А саме, метод невизначених множників Лагранжа, лінійне та нелінійне ціле лінійне програмування тощо. Однак для вирішення цієї проблеми зазвичай використовуються існуючі і перевірені методи.

2.2Перемаршрутизація

Механізм відновлення використовується для боротьби із втратою пакетів (включаючи SDN) у мережі та пошуку резервних маршрутів. В разі використання даного методу знаходяться нові маршрути в мережі, але за умови не перевантаження резервного маршруту. Час пошуку шляху залежить від складності алгоритму та топології мережі, що використовуються OpenFlow, часто в десятих частках мілісекунди.

Існує багато варіацій перемаршрутизації.

- Відновлення доріжки початковим вузлом. Цей механізм є звичайним і реалізується з використанням протоколів маршрутизації.
- Пошук нового маршруту, який обходить елемент, що вийшов з ладу. Важливо те, що лише один вузол мережі шукає новий шлях, тобто початковий вузол шляху. У нашій роботі цей метод не може бути реалізований, якщо початковий вузол насправді не є контролером SDN.
- Захист лінії. Цей захист утримується між двома мережевими пристроями, безпосередньо з'єднаними лінією зв'язку. Обхідні маршрути потрібно знаходити заздалегідь, перш ніж відбудеться збій, і між цими пристроями зв'язок організується заздалегідь, щоб можна було обійти лінію зв'язку у разі відмови. Захист лінії є тимчасовим заходом, оскільки як тільки ви починаєте використовувати байпас, починається процес відновлення. Після перезапуску доступ до резервного маршруту припиняється. Недоліків цієї установки є відсутність гарантії стабільної пропускної здатності. Однак ця установка працює дуже швидко, і час перемикання не перевищує 50 мс.
- Захист вузлів. Цей механізм схожий на попередній, але їх відмінність полягає в тому, що байпас використовується для обходу пристрою, який вийшов з ладу таким чином. Всі інші особливості схожі на особливості захисту лінії, ця установка також є тимчасовою.
- Захист маршруту. Вздовж основної мережевої траси прокладено новий маршрут, який з'єднує кінцеві пристрої, але проходить через комутатори та лінії зв'язку, які не брали участі в основному маршруті. Ця установка є найбільш універсальною, але вона працює повільніше, ніж установка захисту вузлів та ліній.

2.3Резервування

Використовує надмірність для підвищення надійності телекомунікаційних систем та елементів, які входять до певних видів використання. Типи резервного перемикання: структурне, інформаційне, тимчасове та програмне. Інформаційне резервування використовує надмірну інформацію. Тимчасова надмірність - надмірне використання часу. Усі ці типи премикання використовуються в системі поодинці або разом. Нижче наведені всі типи захисного перемикання (Рис 2.2)

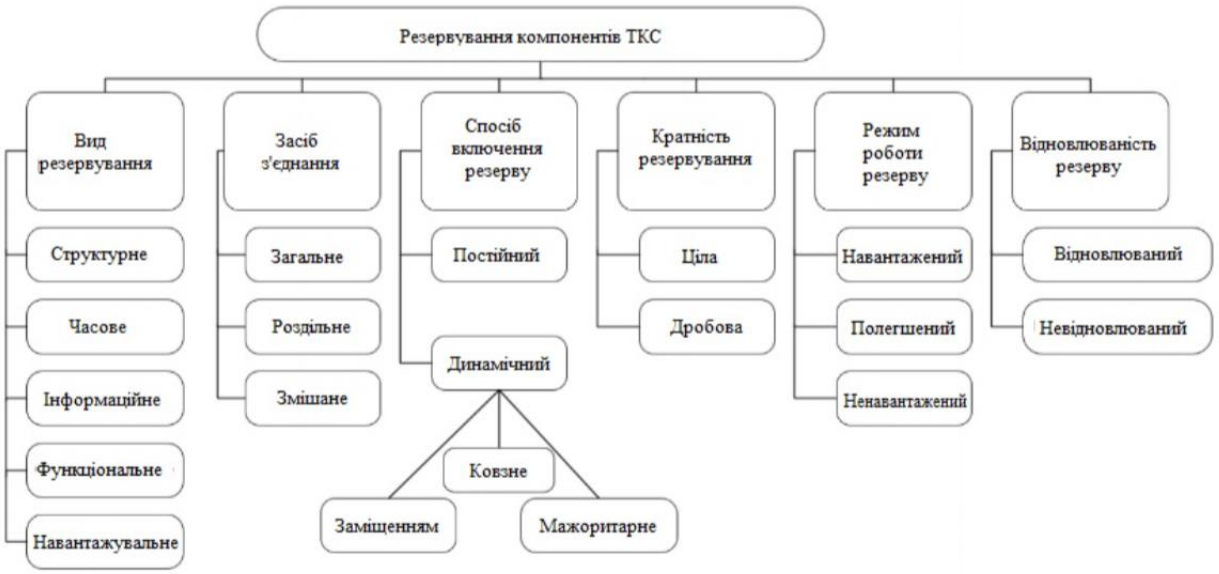


Рис. 2.2 Повна класифікація резервування

За схемою підключення види перемикання поділяються на постійне, динамічне, замісну надмірність та ковзну надмірність як схему включення елементів. В Постійні надлишкові запасні елементи тісно співпрацюють з основними, і це найбільш надійний метод, наведений вище - Риск 2.3. Не існує особливих вимог до конструкції включення додаткових елементів у роботу для постійного резервування у разі відмови.

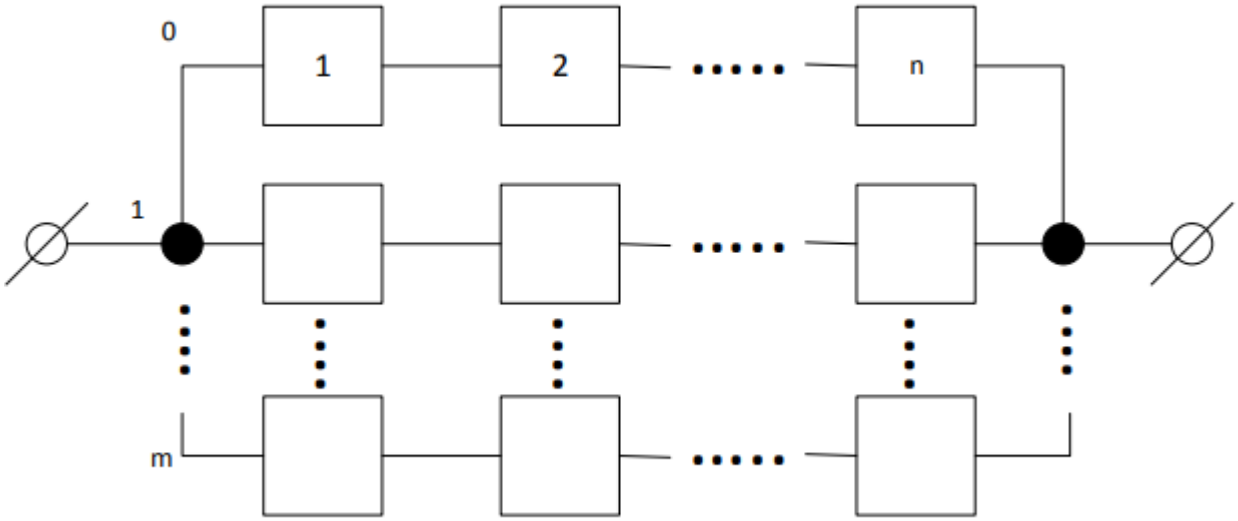


Рис. 2.3 Спільне резервування з постійним резервом

Розділене резервування - це спосіб підвищення надійності шляхом ізоляції елементів системи.

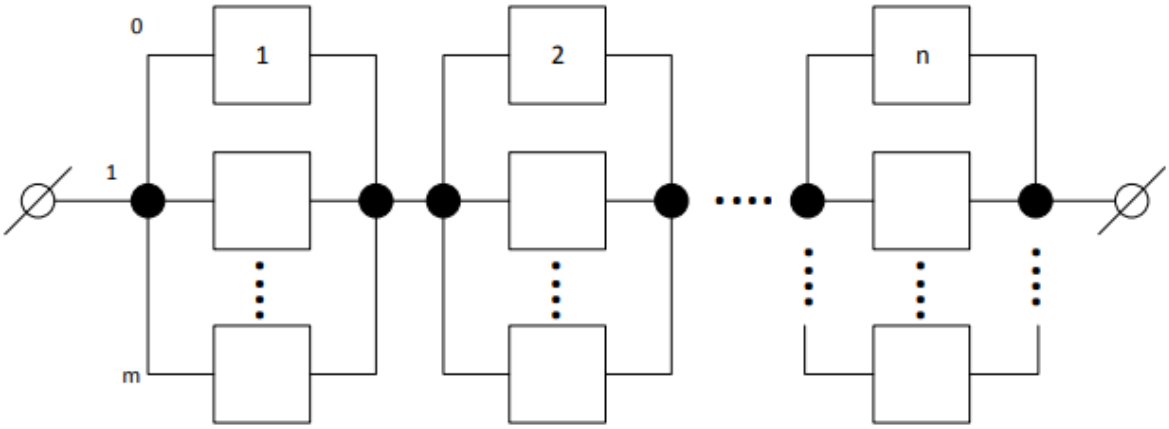
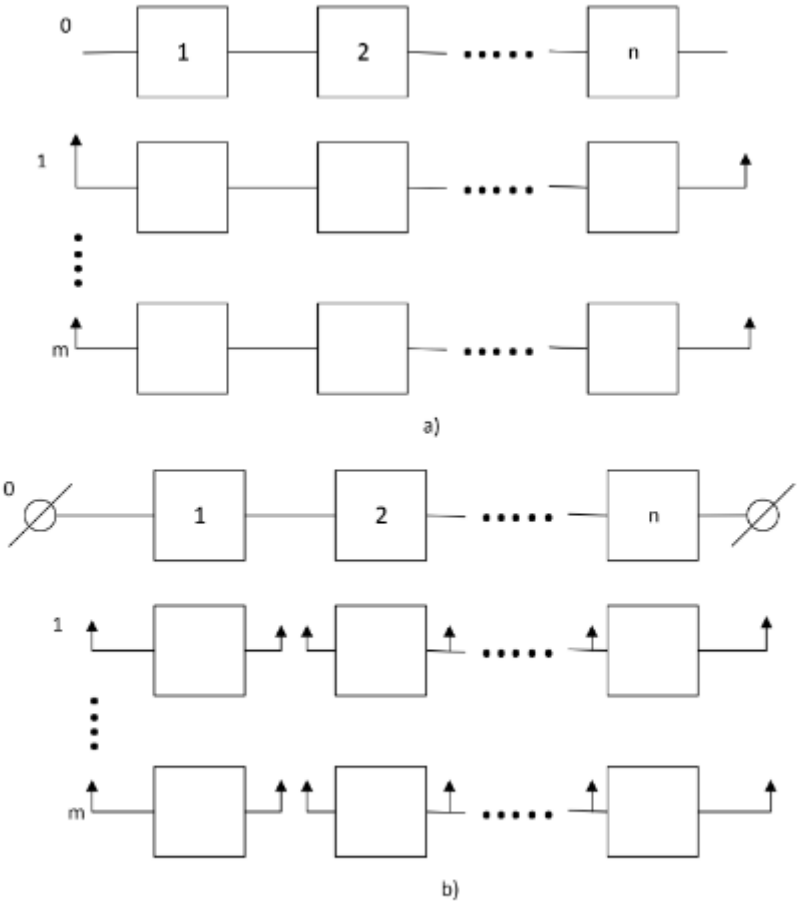


Рис. 2.4 Розділене резервування з постійним резервом

Кратність (або ступінь надмірності) є основним параметром резервного перемикання . Кратність виражається відношенням кількості елементів резерву до основних. Резервування з заміщенням полягає в тому, що функції основної системи надаються запасній тільки в разі відмови основної (Рис 2.5).



a) - загальне резервування; b) - роздільне резервування

Рис. 2.5 Резервування з заміщенням

При використанні ковзного резервування група основних елементів зарезервована одним або кількома запасними елементами. Також можливо замінити елементи, що відмовили, з будь-якої групи в системі (Рис. 2.6).

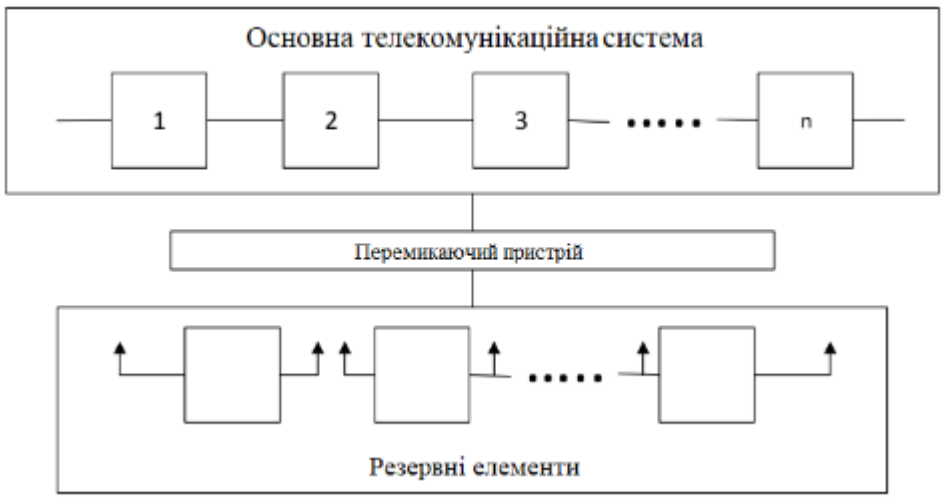


Рис. 2.6 Схема ковзного резервування

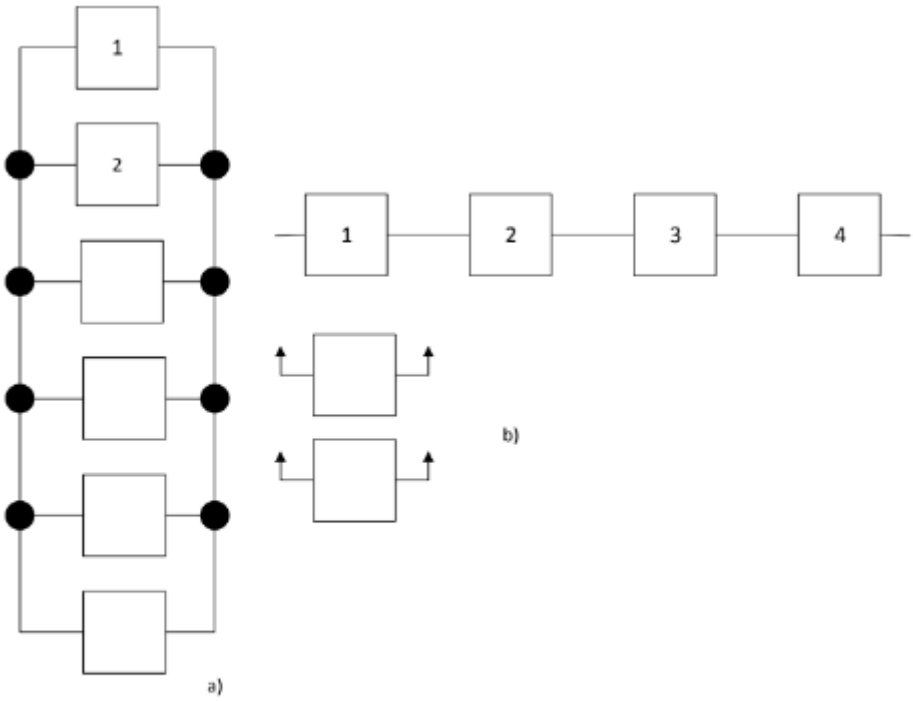
Завданням включення резервування є забезпечення належного функціонування системи після виходу з ладу її елементів. Структурна надмірність (або апаратне забезпечення) передбачає використання системи, елементами якої є базові, додаткові елементи, вузли, пристрої або

Вважається, що замість однієї системи використовується багато подібних систем.

Залежно від режиму роботи розрізняють:

- резерв навантажений - додатковий елемент переважно знаходиться в тому ж режимі роботи, що і заміщуваний - передбачається, що характеристики надійності запасних елементів залишаються незмінними, поки вони знаходяться в резерві та під час використання замість основного, після попередньої несправності;
- резерв полегшений - запасний елемент перебуває в режимі меншого навантаження, ніж основний - передбачається, що характеристики надійності запасних елементів як резерву вищі, ніж їхні, коли вони використовуються замість основних після останньої несправності.

Існує перемикання з цілим та дробовим значенням надмірності. Надмірність позначається літерою *m*.



a) - постійне резервування з кратністю (m=4/2);

b) - роздільне резервування з кратністю (m=2/4)

Рис. 2.8 Резервування з дробовою кратністю

Для резервного копіювання систем, що містять однакові елементи, можна використовувати невелику кількість запасних елементів замість будь-яких основних (ковзне резервування). Полегшений резерв – резервний елемент практично не несе навантаження, коли такий резервний елемент знаходиться всередині. Резервний елемент не повинен відмовляти, тобто має чудову надійність у цей період. Протягом періоду використання цього елемента замість основного після попереднього відмови надійність дорівнює надійності основного.

Особливим випадком надмірності з дробовим значенням надмірності є мажоритарне резервування, яке часто використовується в дискретних інструментах (рис. 2.9).

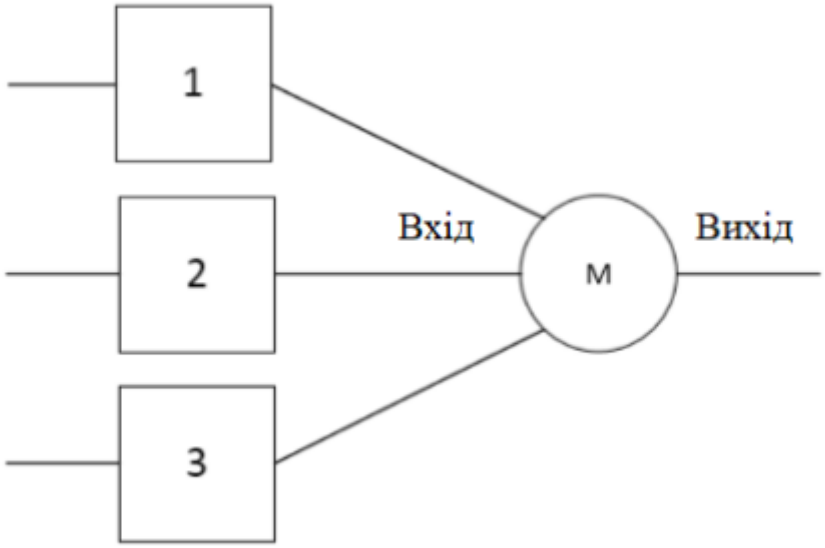


Рис. 2.9 Мажоритарне перемикання

У разі резервування більшості замість одного елемента (каналу) підключаються три однакові елементи, які подаються на виходи. Орган більшості М (елемент голосування). Якщо всі елементи цієї резервної групи працюють, тоді на вхід М надходить три однакові сигнали, і той самий сигнал надходить у зовнішню ланцюг з виходу m. А) постійний надлишковий з кратним (m = 4/2); Б) Відокремте надмірність із множенням Якщо один із трьох запасних елементів виходить з ладу, на вхід М надходить два однакові сигнали (істина) і один сигнал FALSE. Вихід М вказує на сигнал Його внесок відповідає знаку більшості, органу більшості, більшості, що проводить голосування або вибори. Отже, умовою безперебійної роботи з мажоритарним резервуванням є можливість експлуатувати три та будь-які два елементи фактичної мажоритарної системи протягом певного періоду. Комбінований резерв - на зображенні. 1.9 показана група із резервуванням, яка поєднує в собі переваги резервування навантаження (безперервність роботи) та резервування завантаженого навантаження (забезпечує значний виграш у надійності). У цьому випадку два елементи утворюють дублікат групи (завантажений резерв), а третім є ненавантажений резерв. Цей резерв називається спільним.

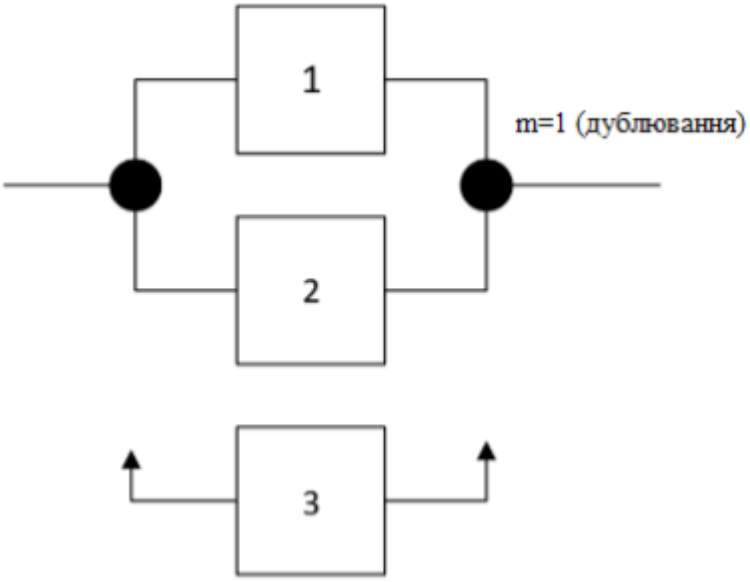


Рис 2.10 Комбінований резерв

Може використовуватися на всіх типах конструкційних надмірних пристроїв, призначених для управління (рис. 2.11).

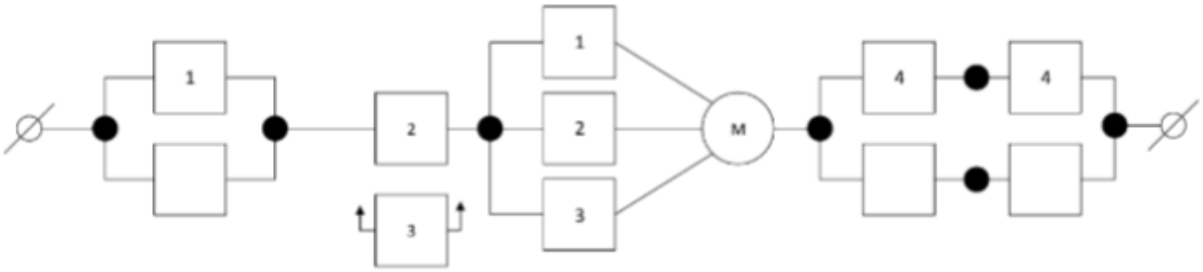


Рис 2.11 Розрахунково-логічна схема структурного резервування

Теоретично введення надмірностей у структуру системи та вибір оптимального режиму може створити будь-яку надійну телекомунікаційну систему. Розглядаючи всі типи скорочень, необхідно зробити деякі практичні висновки: неможливо з економічних причин забезпечити високу надійність із загальним зваженим резервом. Надлишковість елементів повинна дати максимальний ефект. Порівнюючи завантажені та вивантажені резерви між собою, ви можете побачити, що в подібних випадках система є більш надійною, коли система не завантажена завантаженими резервами.

Надійність телекомунікаційної системи безпосередньо залежить від методів резервного копіювання між мережевими пристроями. Комплект технічного обладнання та ліній зв'язку, розроблених спеціально для передачі джерел інформації, називається маршрутом передачі інформації.

Щоб визначити ступінь безпеки, необхідної для даного сегмента мережі, слід враховувати потенціал відмови сегмента мережі та очікуваний вплив на трафік (з точки зору часу відновлення та можливості втрати пакетів).

Потенціал збою в секторі безпеки можна визначити, виходячи з наявної інформації про збій. Початкове значення ймовірності відмови може бути вказане на основі фактичних фактів. Якщо ймовірність відмови відома, необхідно проаналізувати, як відмова впливає на мережевий трафік, тобто визначити "ступінь впливу відмови". Важливим аспектом оцінки впливу відмови є якість обслуговування (QoS), що визначається двома компонентами: часом, необхідним для відновлення, і кількістю втрачених пакетів.

Час для відновлення $T_в$ визначається циклом відновлення иаршруту. Цей цикл може бути заданий такими компонентами:

1) час ідентифікації несправності T_1;

2) Час утримання (якщо потрібно) T_2;

3) Час сповіщення T_3;

4) Час резервування маршруту та сигнал T_4;

5) Час переключити трафік T_5 з активного маршруту на резервний.

Кількість втрачених пакетів $N_{вп}$ дорівнює часу відновлення $T_В$ і швидкості передачі пакетів R, тобто $N_{вп} = RT_в$.

Час, необхідний для виявлення несправності, і час перемикання та перемикання залежить від технології відновлення, що використовується в конкретній системі. Крім того, час встановлення резервних маршрутів (за

умови відмови) залежить від маршруту та методу сигналізації телекомунікаційної системи.

Скорочення часу сповіщення T_3 є, мабуть, ключовим аспектом при розробці методів захисту мереж. Час сповіщення залежить від розповсюдження сигналу відмови між ТР та відстанню D (i, a), яке можна визначити як сегменти мережі (ребра) між вузлом виявлення несправності (вузол А) та суттю, відповідальною за комутатор. (Вузол і).

Розглянемо непотрібні класичні моделі комунікаційних мереж (рис. 1.11) і дамо їм короткий огляд. Фізична топологія мережі складається з вузлів, з'єднаних лініями зв'язку (канали зв'язку, лінії зв'язку). Якщо ми розглянемо процес передачі від джерела до одержувача, представлено поняття "маршрут". Існують основні (робочі) шляхи та резервні копії. Розділ листівки з кількома посиланнями називається "розділом". Визначення поняття "сегмент" можна прийняти як узагальнення визначення понять "робочий шлях" та "посилання".

На рисунку 1.11 показана модель захисту. Тут кожне посилання захищене окремо (локальна безпека). Цей метод є дуже обчислювальним

Забезпечує ефективність, швидку маршрутизацію, простий і масштабований, але вимагає великих мережевих ресурсів.

Захист шляху (рис. 1,11b) здійснюється за допомогою наскрізного шифрування, як у кінцевих пристроях користувачів (іноді їх називають глобальним захистом). Тут мережеві ресурси використовуються економно, але розрахувати маршрут від кінця до кінця складніше. Існує два варіанти цього захисту:

1) альтернативний маршрут, який використовує одну або кілька ланок в робочому контурі;

2) Альтернативний маршрут, який не відповідає основному маршруту в жодному з посилань.

Другий варіант можна розглянути, якщо збої з’являються на будь-якому зв’язку на основному маршруті (тоді для кожного випадку відмови в першому.

Альтернативі доведеться знайти альтернативний шлях). У разі виходу з ладу будь-якого зв’язку основного маршруту, якщо використовується другий варіант, відновлення може бути розпочато негайно, без зазначення проблеми, що є. Рис 1.11c та розділи 1.11 показують моделі безпеки сегментів (багатоланкові розділи). Модель на малюнку 1.11d відрізняється тим, що вона демонструє захист "накладання", що дозволяє обійти невдалі вузли (крім вузлів джерела та приймача).

На рисунку 2.12 показаний захист кільця на основі П-циклу.

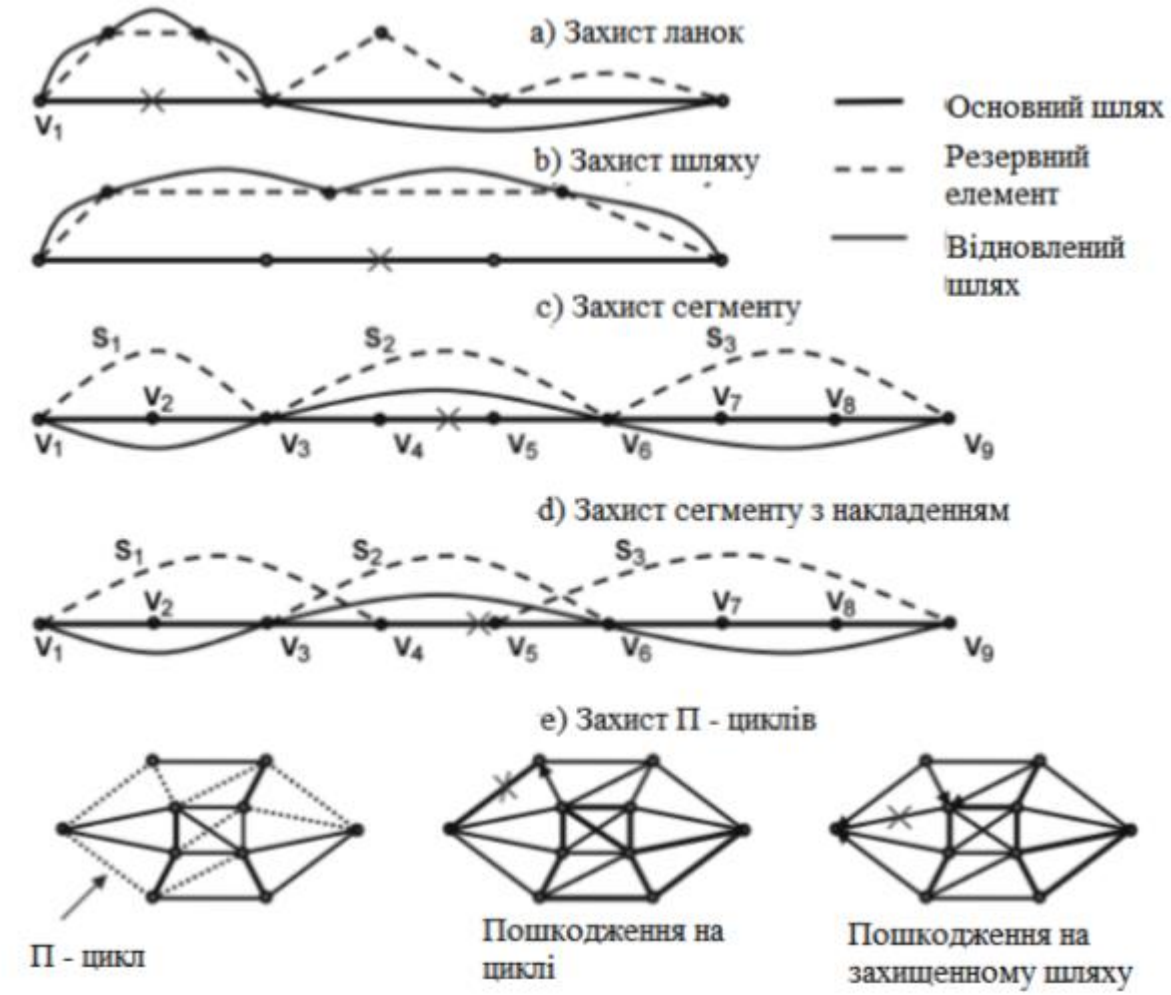


Рис 2.12 Модулі резервування на основі П-циклу

Суть занедбаності, заснованої на П-циклі, полягає у розміщенні замкненого циклу або циклу у сильно зв’язаній топологічній структурі з початковим розрахунком пропускної здатності, яка може бути застосована в результаті відмови мережі зв'язку. У прикладі моделей, зображених на рисунку 2.12, розглядається лише вибір зони безпеки (або зони безпеки). Далі ми зупинимось на відомих методах використання ресурсів пропускної здатності, таких як 1+1, 1:1, M:N, які можна використовувати як для параметрів безпеки шляху, так і для варіантів безпеки.

Захист маршруту 1+1. Дані передаються спільною роботою та резервними маршрутами. Найкращий знак був призначений на етапі прийому. Робочий та резервний шляхи розділені.

Захист ланки 1+1. Принцип роботи однаковий з точки зору безпеки маршруту, але сканується лише одне посилання або вузол, який не працює, а не весь маршрут.

Захист ланки 1:1. Перед відмовою на робочий шлях надсилаються лише дані. Вторинний трафік може передаватися резервним шляхом. У разі виходу з ладу робочого каналу передача вторинного трафіку зупиняється, а дані передаються з резервного каналу, який працює. Якщо несправність робочого шляху усунена, можливі наступні варіанти:

1) Повернення на роботу з резервного маршруту;

2) Після вирішення проблеми трафік залишається на резервному маршруті, тоді як працівник виконує резервну роботу. Перевагою першого варіанту є пріоритет використання більш надійного шляху руху, яким зазвичай є працівник. Недоліком є необхідність комутації, яка виконується пристроєм з K_r <1.

Захист тракту 1:1. Принцип роботи однаковий з точки зору безпеки Посилання 1: 1, але він забезпечує обхід всього пошкодженого шляху. У деяких випадках використовується так звана групова безпека (спільна) або безпека M: N, що є узагальненням безпеки форми 1: 1, (M = 1, N = 1).

Захист тракту M: N. Шляхи завдань і резерву організовані на відмову (N працює, M резерв, N ≥ M). У разі збою посилання дані передаються на резервне посилання, але якщо пошкоджено більше M посилань, вторинний трафік буде втрачений. Найчастіше використовується варіант M: N, що відповідає M = 1 (1: N).

Захист шляху M:N. Принцип дії однаковий з точки зору захисту, але він забезпечує об'їзд усього маршруту. Цей спосіб бронювання є найбільш популярним завдяки низькій вартості та гнучкості. Однак цей варіант дуже важко оптимізувати, особливо якщо є потреба у використанні рослин, які зосереджують увагу.

Процедури резервного копіювання, обговорені вище, можна використовувати разом із процесом відновлення.

2.4Висновки до розділу

В даному розділі були розглянуті основні підходи для забезпечення надійності в програмно-конфігурованих мережах, розглянуто питання розміщення SDN-контролерів з метою максимізації надійності програмно-конфігурованих мереж. Надається метрика, названа відсотком втрат в тракті управління, щоб охарактеризувати надійність мереж управління SDN. Формулюється проблема розміщення контролера, пропонується кілька алгоритмів розміщення, які допомагають вирішити цю проблему. Завдяки моделюванню з величезною кількістю топологій, наочно демонструється як кількість контролерів і їх розміщення впливають на надійність мереж управління.

3. МЕТОД ПОБУДОВИ ХМАРНОГО СЕРВІСУ НА ОСНОВІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

3.1. Опис тестового стенду для оцінки

Предметом дослідження є мережа, яка повинна утримувати хмарне сховище і побудована за принципами технології SDN, де використовуються комутатори з малим функціоналом, а всі головні завдання управління, маршрутизації та іншого виконують контролери SDN.

Першочергова задача такої мережі – забезпечення доступу користувачів до хмарного сховища та файлів, які зберігаються в ньому. Усі запити користувачів йдуть до ЦОДу, який обробляє дані та проводить рендер веб-сторінки на стороні сервера після чого відправляє клієнту необхідні дані.

Технологію SDN було обрано за легкість у масштабуванні та через необхідність побудови складної топології для хмарних сервісів, побудованих за принципом віртуалізації мережевих функцій. Мережа SDN також не є залежною від постачальників обладнання і не має ряду проблем, пов’язанних із сумісністю певних приладів, що необхідні для функціонування складної корпоративної мережі. Оскільки надійність та час відновлення є ключовими для бізнесу, то ключовою задачею є забезпечення надійності (відмовостійкості).

На рисунку 3.1 представлено узагальнену схему досліджуваної мережі до переходу на SDN.

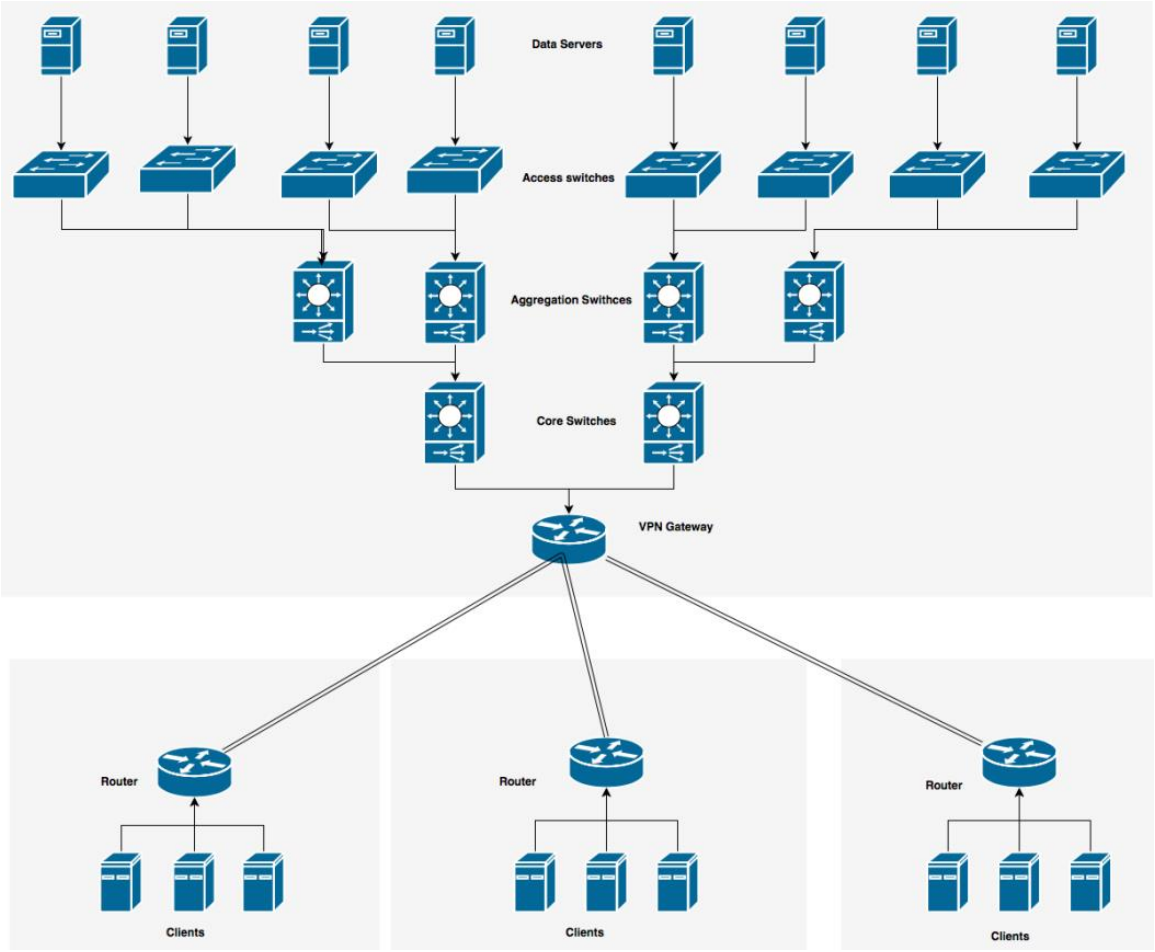


Рис 3.1 Узагальнена схема топології мережі хмарного сервісу

Забезпечити надійність SDN мережі можна на кількох рівнях: на нижніх рівнях моделі OSI за допомогою інфрастуктури мережі та на прикладному рівні за допомогою спеціальних протоколів відновлення. В першу чергу варто розглянути інфраструктуру мережі і відштовхуючись від неї розглянути подальші методи забезпечення надійності.

Для побудови заданої системи знадобиться наступний набір інструментів та ПЗ:

- VirtualBox – гіпервізор
- NEC ProgrammableFlow Controller Software (PFC) – SDN-контролер, який є доступним по тріальній ліцензії
- MiniNet OFV – ПЗ для розгортання топології SDN мережі

- Open vSwitch (OVS) 2.3 – ПЗ для контролю маршрутизаторів, які працюють за протоколом OpenFlow
- Веб-сервіс Apache Web Server
- MariaDB – база даних
- Бібліотека PHP
- OwnCloud

3.2. Топологія мережі

Розглянемо окремий SDN контролер для оцінки забезпечення надійності системи (Рис 3.2).

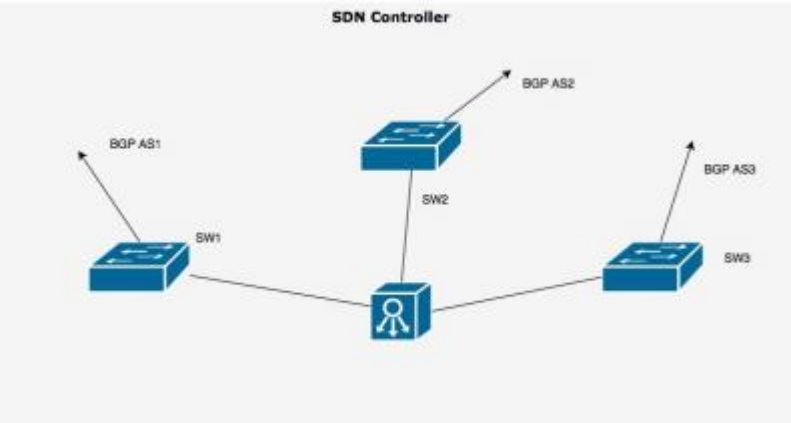


Рис 3.2 Сегмент досліджуваної SDN мережі

Основою аналітичних методів є теорія випадкових процесів (в нашому завданні - марковських). За допомогою однорідних марковських процесів з кінцевим числом станів описується система при обмеженнях. Час перебування в одному стані розподілено по показовому закону. У справжнім методах розрахунку надійності приймається допущення, що відмови елементів незалежні, і система потрапляє в стан відмови при відмові певного числа елементів.

На рисунку 3.2 представлена схема сегменту досліджуваної мережі. Контролер SDN - це самий технічно складний пристрій мережі, якщо відмовить контролер, то станеться обрив зв'язку з зовнішніми мережами. Для

зв'язку з зовнішніми мережами використовуються маршрути через автономні системи BGP. Для дослідження впливу резервування SDN контролера, вводимо допущення при обчисленнях:

- при недоступності одного з маршрутів смуги пропускання, що залишиться, буде достатньо для задоволення потреб;
- розглянута мережа вважається непрацездатною при відмові контролера SDN, а також при відмові всіх комутаторів;
- при роботі мережі, в один момент часу може відмовити тільки один сервер;
- час відновлення контролера набагато більше часу відновлення комутатора, тому відновлення комутатора відбувається непомітно при одночасному відновленні контролера.

Для даної мережі складемо список станів:

1) відмови відсутні;

2) відмова одного комутатора;

3) відмова двох комутаторів;

4) відновлення комутатора при робочому сервері;

5) відновлення двох комутаторів при робочому сервері;

6) відмова сервера;

7) відмова комутатора і сервера;

8) відмова сервера і двох комутаторів;

9) відновлення контролера;

10) контролер з кластера серверів непрацездатний.

При резервуванні система непрацездатна в стані 9 і 10, при відсутності резервування система вважається непрацездатною в станах 6, 7, 8, і 9.

При використанні резервування контролера SDN з'являється новий стан, при якому система переходить в стан №10 - контролер з кластера серверів непрацездатний. При використанні нового стану і утворилися логічних переходів між новим станом і вихідними станами системи виходить новий граф станів мережі. Всі переходи між станами відбуваються з заданими в таблиці 2.4 інтенсивностями.

Таблиця 3.1 Інтенсивіність переходів між станами

Позначення	Значення	Опис
λ_{sw}	1/516593 $ч^{-1}$	Інтенсивіність відмов комутаторів
λ_{sr}	1/1700000 $ч^{-1}$	Інтенсивіність відмови сервера
λ_0	1/0,5 – 1/24 $ч^{-1}$	Інтенсивіність виявлення відмов
μ	1/4 $ч^{-1}$	Інтенсивіність відновлення комутатора
μ_m	1/6 $ч^{-1}$	Інтенсивіність відновлення контролера

Складемо рівняння для визначення ймовірностей кожного стану марковського процесу для даної системи без резервування і з застосуванням резервування (диференціальні рівняння А.Н. Колмогорова). Система диференціальних рівнянь Колмогорова для нерезервованої системи має вигляд

$$\begin{cases} \frac{dP_1(t)}{dt} = -(2\lambda_{sw} + \lambda_{sr})P_1 + \mu P_3 + \mu P_4 + \mu_m P_8 \\ \frac{dP_2(t)}{dt} = \lambda_{sw}P_1 - (\lambda_0 + \lambda_{sr})P_2 \\ \frac{dP_3(t)}{dt} = \lambda_{sw}P_1 - (\lambda_0 + \lambda_{sr})P_3 \\ \frac{dP_4(t)}{dt} = \lambda_0 P_2 - \mu P_4 \\ \frac{dP_5(t)}{dt} = \lambda_0 P_3 - \mu P_5 \\ \frac{dP_6(t)}{dt} = \lambda_{sr}P_1 - (2\lambda_{sw} + \lambda_0)P_6 \\ \frac{dP_7(t)}{dt} = \lambda_{sr}P_2 + \lambda_{sw}P_6 - \lambda_0 P_7 \\ \frac{dP_8(t)}{dt} = \lambda_{sr}P_3 + \lambda_{sw}P_6 - \lambda_0 P_8 \\ \frac{dP_9(t)}{dt} = \lambda_0 P_6 + \lambda_0 P_7 + \lambda_0 P_8 - \mu_m P_9 \end{cases}$$

Після отримання чисельних значень коефіцієнтів готовності, можна побудувати графік залежності коефіцієнта готовності від інтенсивності виявлення відмов, при використанні резервування контролера і для нерезервованої системи (Рис. 3.2).

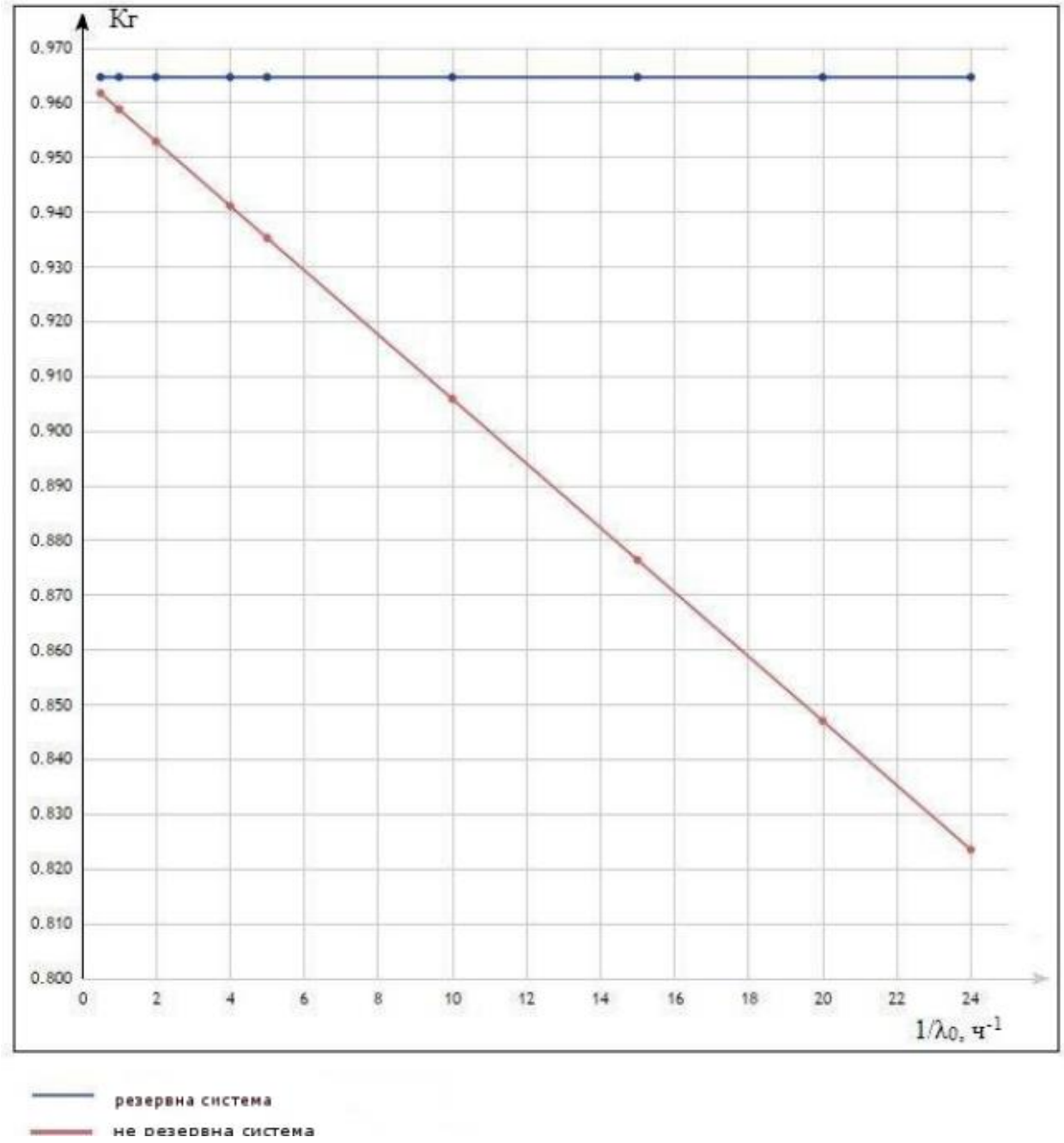


Рис 3.2 – Графік залежності коефіцієнта готовності від інтенсивності виявлення відмов

Роблячи висновок по результатам розрахунку і наочному уявленню (рис. 3.2), можна сказати, що:

– при однакових числових значеннях, коефіцієнт готовності при використанні резервування вище ніж у нерезервованої системи;

– при використанні резервування $К_г$ менш залежний від інтенсивності виявлення відмов.

Виходячи з вищеописаних результатів можемо узагальнено зобразити проектовану мережу підприємства наступним чином (рис. 3.3):

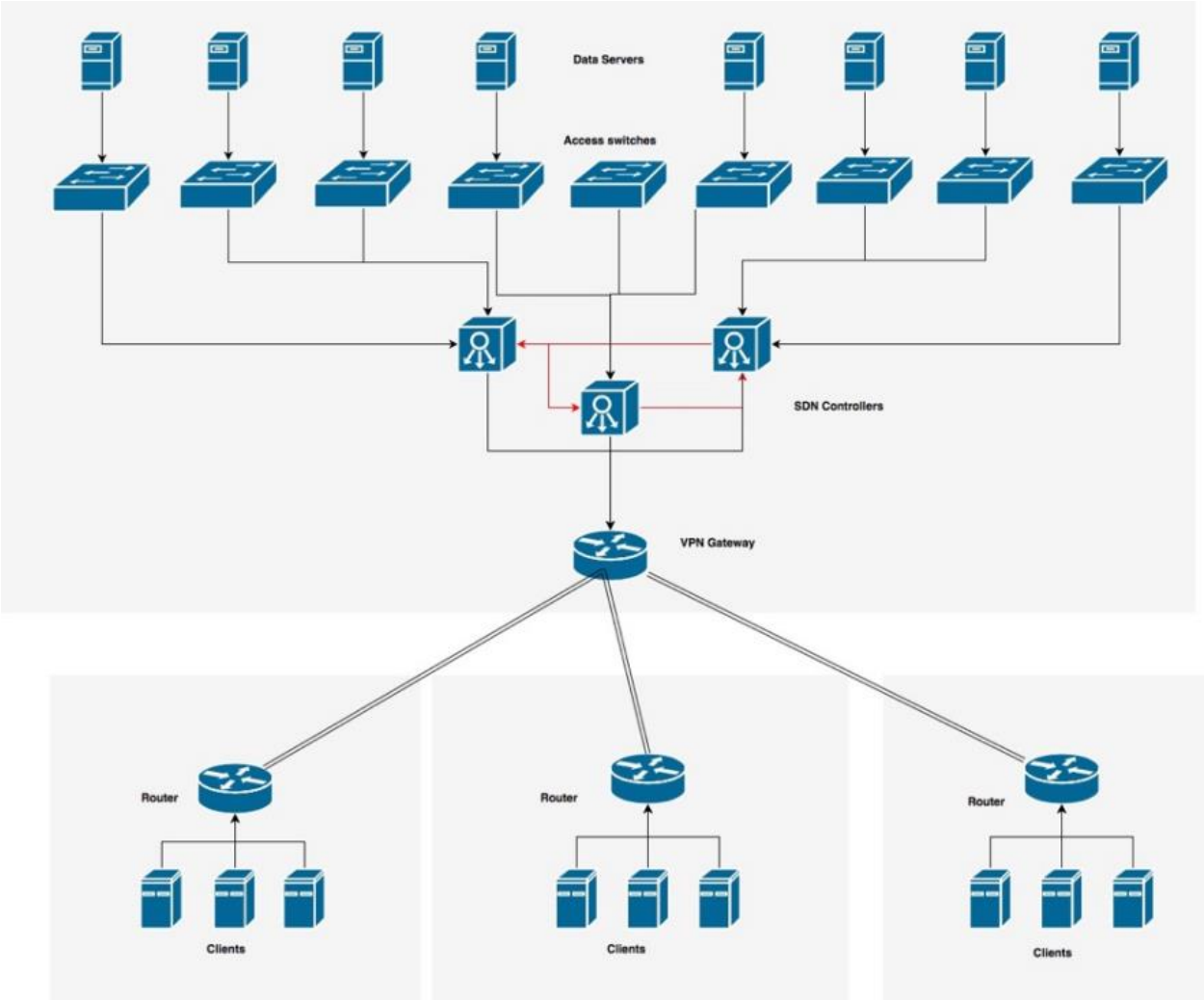


Рис 3.3 Модифікована структура мережі

3.3.Побудова описаної системи

В якості SDN-контролера буде використано умовно безкоштовне програмне забезпечення NEC ProgrammableFlow Controller Software (PFC). Воно доступне по тріальній ліцензії на сайті розробника. Фактично дане ПЗ є операційною системою на ядрі Linux.

Для побудови і конфігурації топології мережі, що буде складатися з комутаторів, що працюють за протоколом OpenFlow та хостів Linux, буде

використано Mininet 2.2.0. Mininet – це теж ОС на основі ядра Linux так само, як і в випадку з PFC.

Як тільки Mininet буде запущено потрібно видалити вбудовану в неї версію Open vSwitch (OVS), яка являє собою ПЗ, що контролює OpenFlow комутатори. Після видалення встановленої версії необхідно замінити її OVS версії 2.3.

Після того, як з попереднім пунктом буде закінчено, необхідно зконфігурувати Mininet таким чином, щоб він не запускав OVS-контролер за замовчуванням, адже замість нього буде використано SDN-контролер NEC ProgrammableFlow Controller Software. Також необхідно налагодити OVS таким чином, щоб він запускався разом з системою Mininet.

Як тільки Mininet буде налаштовано переходимо до налагодження нашого SDN-контролера. При зміні його конфігурації необхідно зупиняти всі мережеві процеси, і після внесення змін запускати знову. За замовчуванням PFC має статичну IP-адресу, тому перш за все необхідно зконфігурувати контролер так, щоб він був в тій самій IP-мережі, що і Mininet.

Нарешті можна побудувати описану в попередньому пункті мережу, використовуючи інструменти, що надає Mininet. Для цього необхідно змінити початкову конфігурацію мережі (Рис. 3.4)

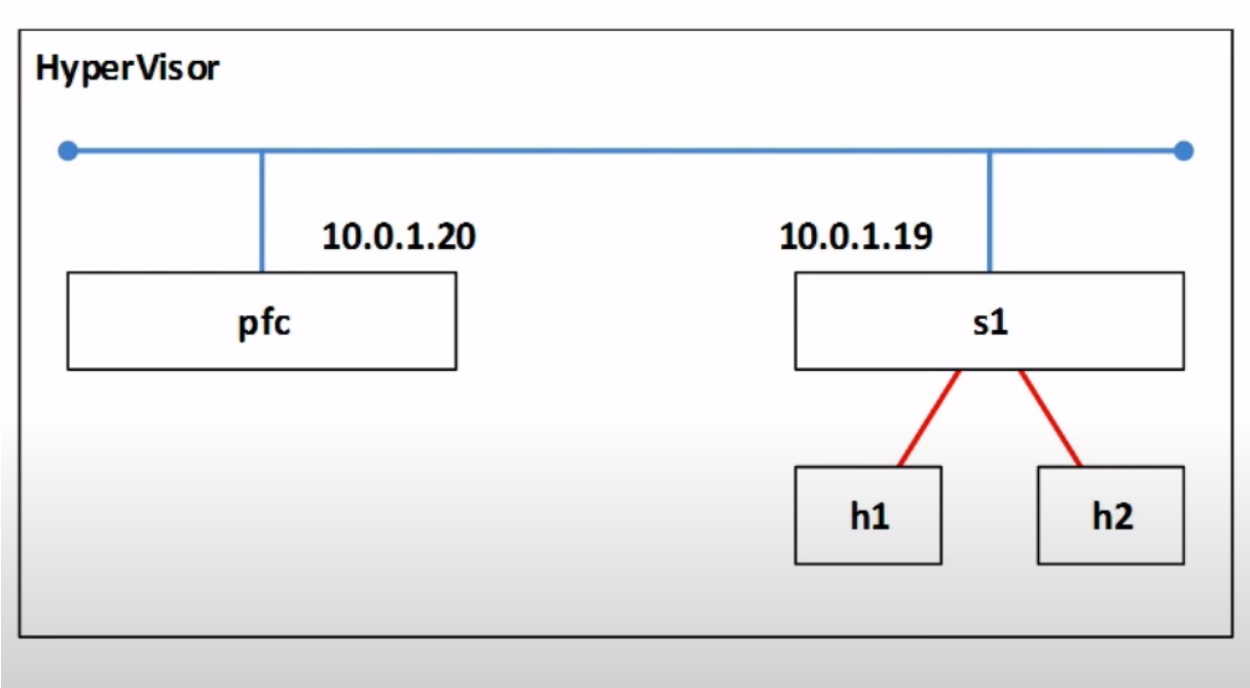


Рис 3.4 Схема конфігурації мережі Mininet за замовчуванням

Після зміни файлу конфігурації мережі можна переходити до побудови хмарного сховища на основі даної мережі.

Перед тим як почати встановлення OwnCloud треба встановити деякі додатки, а саме **Apache**, **MySQL** та **PHP**.

Починаємо з оновлення пакетів. Далі встановлюємо Apache Web Server. Після завершення інсталяції вмикаємо автозапуск сервісу при старті системи та запускаємо сам сервіс. Далі в правилах файрволу треба зробити відповідні зміни, щоб дозволити запити до серверу.

Переходимо до встановлення MariaDB. MariaDB – це гілка БД MySQL. Аналогічно з сервісом Apache переводимо MariaDB в режим автозапуску та вперше запускаємо сервіс. Окремою командою перевіряємо чи MariaDB працює.

Тепер встановлюємо PHP. Для цього знадобиться підключити репозиторій Remi. Далі за допомогою yum-utils вмикаємо remi-repository. Після успішного завершення попередніх кроків шукаємо список доступних PHP модулів, які доступні для завантаження. В списку будуть показані

декілька доступних версій. Нас цікавить 7.2, оскільки Owncloud працює саме на ній і на даний момент не підтримує більш нові версії. Вводимо команди:

Нарешті, встановлюємо PHP, PHP-FPM(FastCGI Process Manager) та всі асоційовані модулі та переводимо всі додатки в режим автозапуску, а також змінюємо конфігурацію для того, щоб Apache міг виконувати команди написані на PHP за допомогою PHP-FPM після чого перезапускаємо веб-сервер.

Для остаточної перевірки заходимо з браузеру віртуальної машини за адресою 192.168.137.26/info.php (де 192.168.137.26 – ip-адреса нашого серверу). Оскільки Owncloud є php-додатком, треба впевнитися, що на ВМ встановлений та працює PHP.

Тепер треба створити БД для Owncloud. Для цього входимо в MariaDB і створюємо користувача для бази даних. Надалі працюємо з запитами SQL.

CREATE DATABASE owncloud_db;

GRANT ALL ON owncloud_db.* TO ‘owncloud_user’@’localhost’ IDENTIFIED BY ‘пароль’;

FLUSH PRIVILEGES;

EXIT;

Переходимо до завантаження Owncloud. На даний момент остання актуальна версія – 10.4.1. Після завантаження цей архів треба розпакувати: Після цього надаємо веб-серверу Apache права доступу та читання файлів і папок Owncloud.

Необхідно змінити деякі дані конфігурації відносно тих, що встановлені за замовчуванням, для того, щоб Apache Web Server забезпечував роботу Owncloud. Отже, створюємо та відкриваємо для редагування конфіг-файл для Owncloud і вводимо наступне:

Alias /owncloud “/var/www/owncloud/”

```
<Directory /var/www/owncloud/>
    Options +FollowSymLinks
    AllowOverride All
    <IfModule mod_dav.c>
        Dav off
    </IfModule>
    SetEnv HOME /var/www/owncloud/
SetEnv HTTP_HOME /var/www/owncloud/
</Directory>
```

Зберігаємо і виходимо. Для того, щоб зміни набули чинності, треба перезавантажити веб-сервер та впевнитись, що після перезавантаження він запустився і працює.

Також необхідно, щоб SELinux дозволяв веб-серверу Apache працювати з файлами в папках Owncloud.

Тепер, коли всі компоненти встановлені і запущені, залишились останні кроки – налаштувати сам Owncloud для роботи.

Відкриваємо браузер віртуальної машини та переходимо за посиланням http://192.168.137.26/owncloud (де 192.168.137.26 - ip-адреса нашого серверу) У випадку, якщо не буде вистачати певних php-модулів на сторінці відобразиться повний список модулів, які необхідно встановити.

Після чого перезавантажуємо веб-сервер. Оновлюємо сторінку в браузері та вводимо дані користувача та натискаємо Storage & database нижче. У полі Configure the database обираємо MySQL/MariaDB та вводимо нижче дані, які були задані раніше (Рис 3.5).

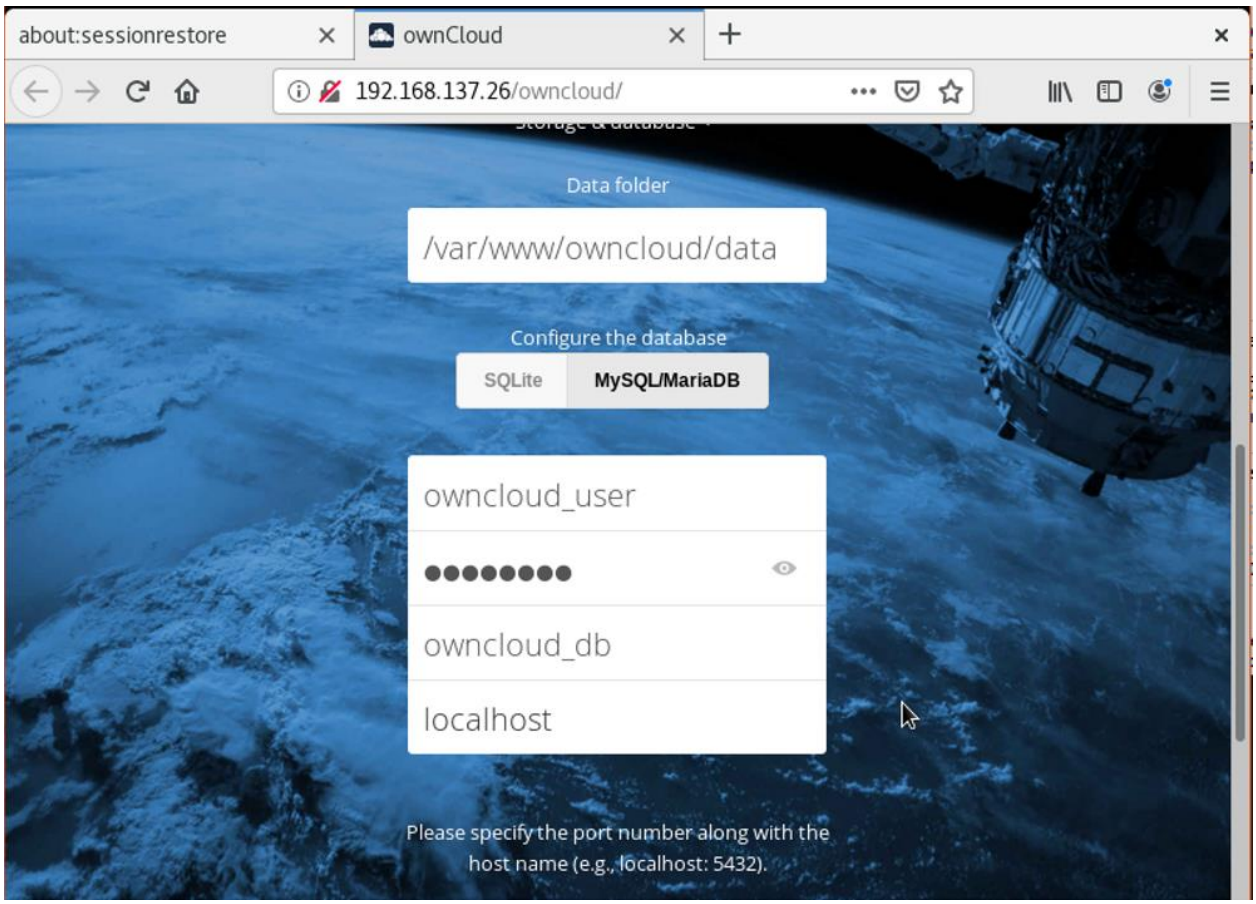


Рис 3.5 Екран входу користувача Owncloud

Нижче натискаємо Finish Setup. Далі вводимо логін/пароль, які щойно задали і бачимо інтерфейс Owncloud, готовий до використання (Рис. 3.6).

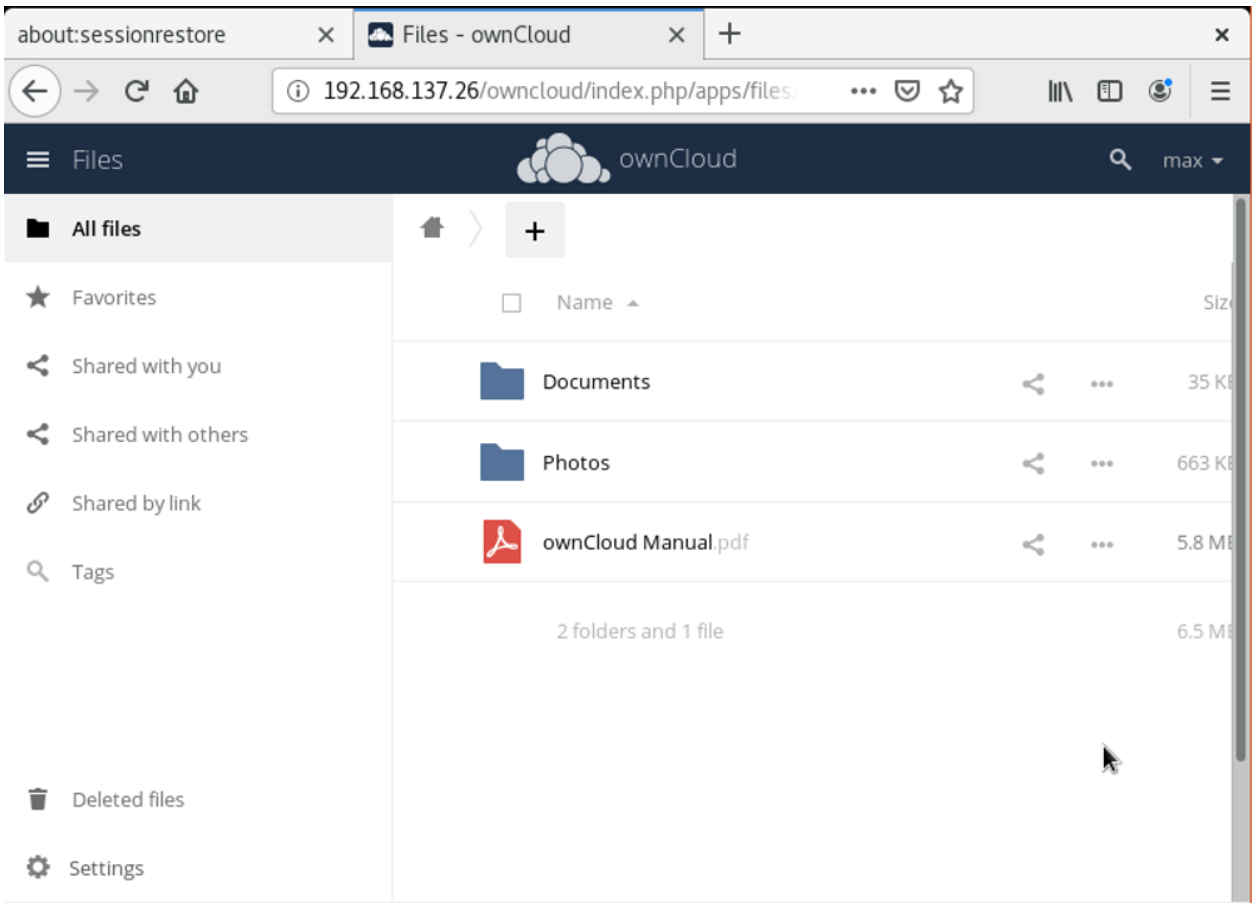


Рис 3.6 Перегляд збережених файлів користувача Owncloud

На даний момент ми зайшли в сховище у ролі адміністратора. Це можно перевірити перейшовши у вкладинку Users (правий верхній вугол). Тут можна подивитися список усіх користувачів, що мають доступ до сховища, видати та створити нові ролі (групи) та загалом адмініструвати користувацьку базу.

Для прикладу створимо користувача *tester1*, створимо нову групу *user* і додамо до цієї групи щойно створеного користувача. Після того, як натиснемо на кнопку Create, оновимо сторінку і побачимо, що користувача було додано до списку. Користувачу *max*, що належить до групи *admin*, дамо керування над групою *user*.

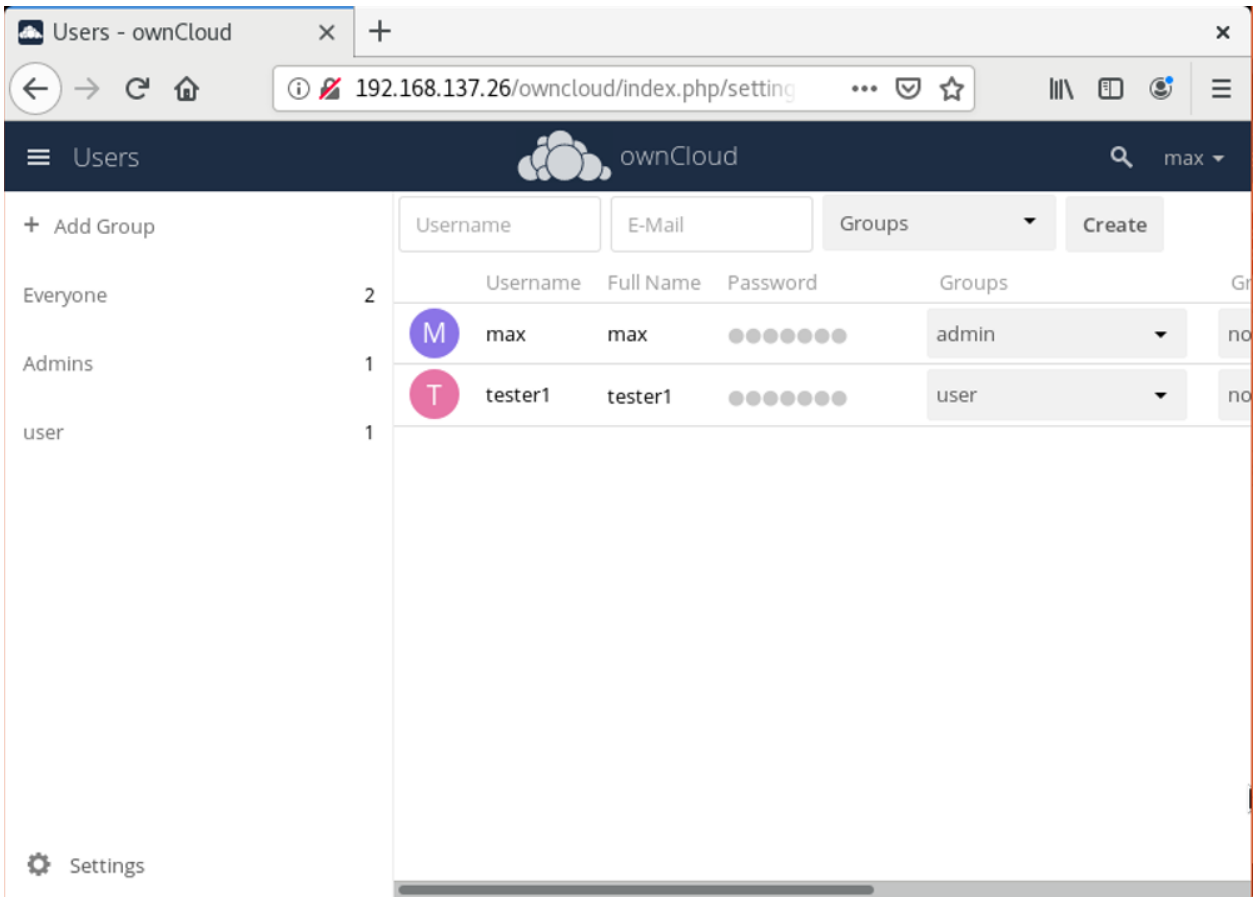


Рис 3.7 Список створених користувачів Owncloud та їх ролі

В нас одразу є 2 папки з файлами та один окремий файл, які ми маємо побачити, коли підключимося до сховища від імені іншого користувача.

Перевіримо роботу ownCloud за допомогою іншої ВМ. Для доступу до сховища у ролі звичайного користувача нам знадобиться встановити спеціальний додаток – ownCloud Desktop Client. Для цього запускаємо іншу ВМ і встановлюємо клієнт.

В ході установки погоджуємось на всі «запитання» та запускаємо клієнт. Після вдалої установки і запуску клієнта вводимо ip-адресу нашого серверу і переходимо далі до вікна авторизації. Вводимо дані раніше створеного користувача tester1 та в наступному вікні залишаємо все за замовчуванням. Після недовгого процесу синхронізації побачимо наступне. Звернемо увагу, що нам доступні папки та файли, що були у сховищі.

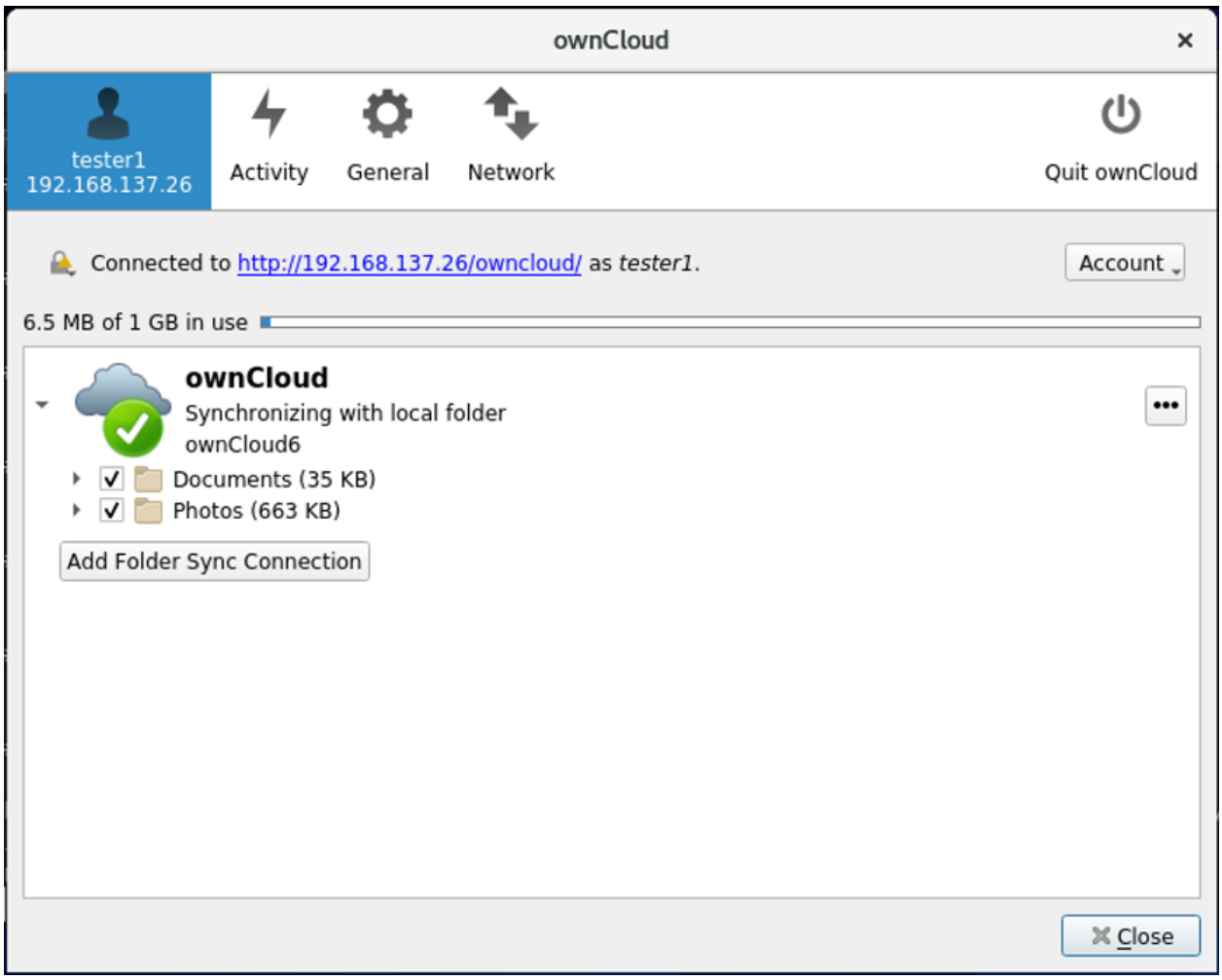


Рис 3.8 Екран користувача ownCloud Desktop Client

Їх же ми можемо побачити, якщо в ВМ перейдемо в папку, що була створена в процесі авторизації і синхронізації. В даному випадку це max1/Home/ownCloud6/. В ній бачимо всі ті ж самі файли.

Перевіримо чи працює синхронізація в протилежному напрямку. Додамо на нашій клієнтській машині в папку Photos новий файл – splash.png. Як тільки це буде зроблено у встановленному нами клієнті у вкладці Activity побачимо, що файл було завантажено на сервер (також видно, що з серверу на клієнтську ВМ раніше були завантажені вищезгадані папки і файли)

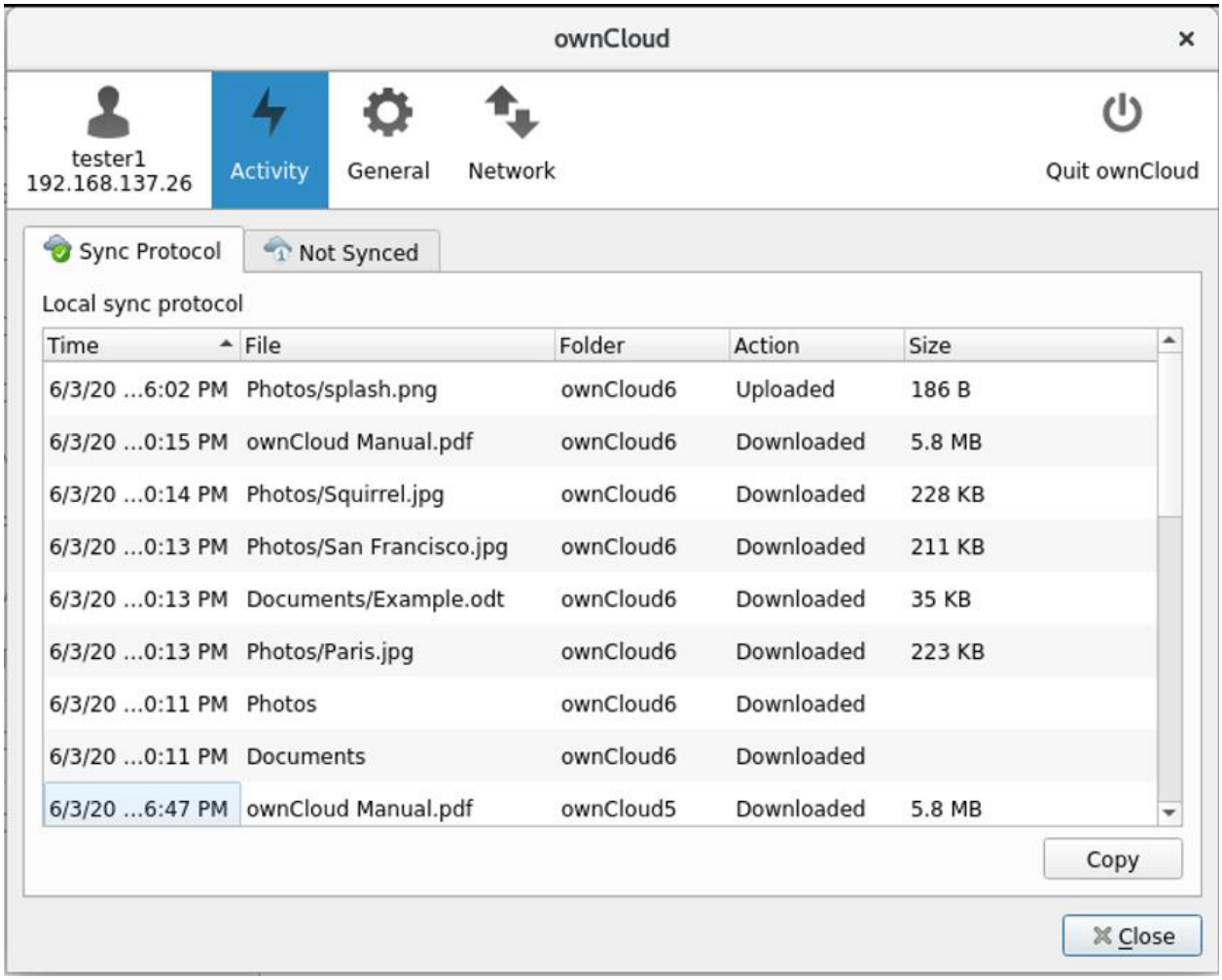


Рис 4.5 Процес синхронізації ownCloud Desktop Client

З серверної ВМ бачимо, що файл splash.png дійсно було завантажено.

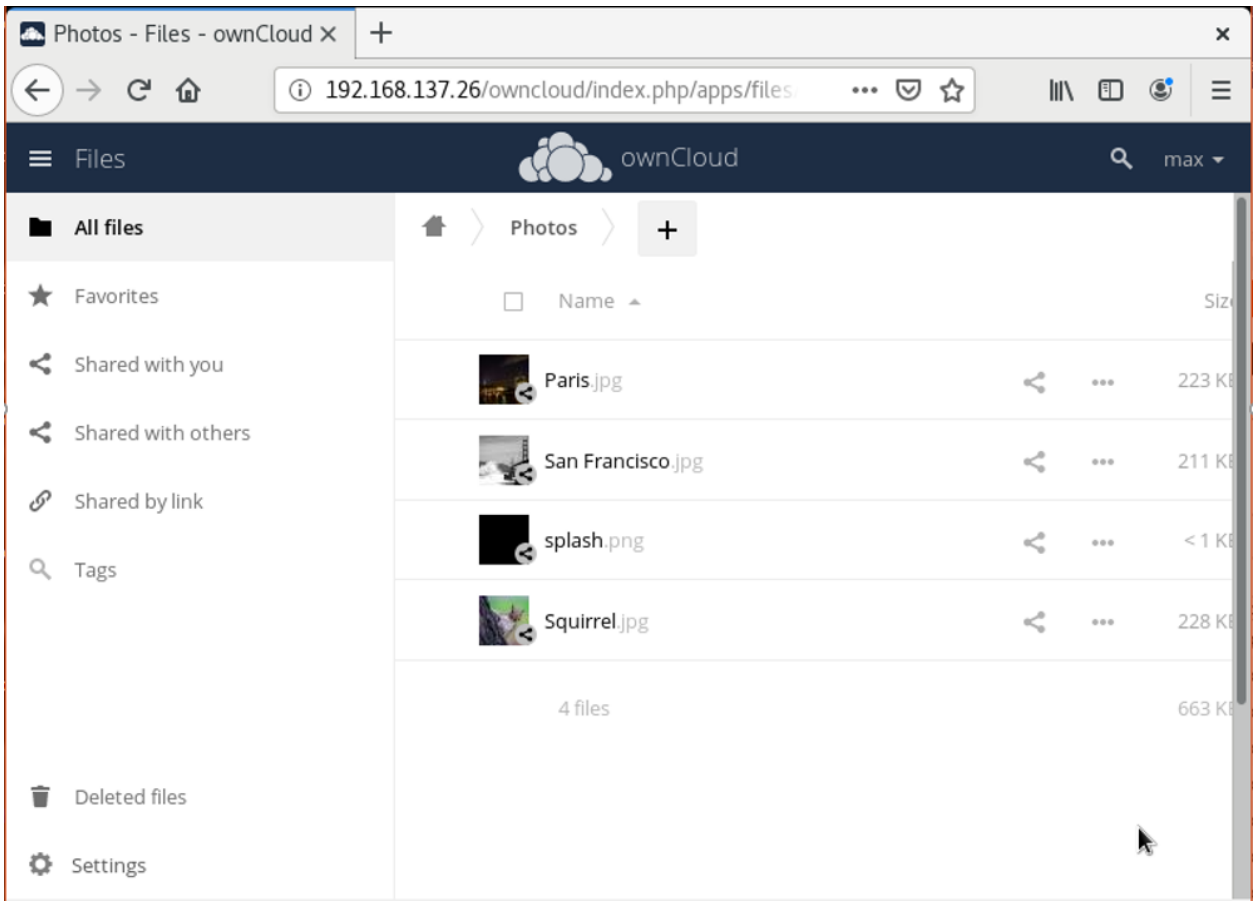


Рис 4.6 Результат синхронізації ownCloud Desktop Client

3.4. Висновки до розділу

В даному розділі було описано підхід до побудови мережі на основі SDN для використання в якості основи для хмарного сервісу. Було розроблено топологію мережі з врахуванням необхідності в підвищенні відмовостійкості системи і описано алгоритм розрахунку з якого випливає остаточно форма результуючої мережі.

За допомогою програмного забезпечення вільно доступного в мережі Інтернет, а також за рахунок умовно безкоштовного ПЗ PFC було побудовано описану мережу та запущено хмарне сховище.

Після цього було проведено тестування системи на предмет коректної роботи.

4. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1. Опис ідеї проекту

Бізнес SDN додатків не є новим, незважаючи на молодість технології. Існують приклади вже успішних застосувань технології SDN у великих корпоративних мережах. Розроблений програмний продукт призначений для демонстрації ідеї хмарного сервісу на основі SDN з забезпеченням відповідного рівня відмовостійкості і на даному етапі розробки не може бути представлений як повноцінний продукт. Проте, за умови подальшого розвитку проекту, можна спланувати основні ідеї, положення та скласти стратегію дій для розробки ефективного, конкурентоспроможного додатку.

В рамках магістрського проекту, можна реалізувати наступну ідею: мережа з топологією, описаною в попередньому розділі, яка побудована на основі SDN з забезпеченням високого рівня відмовостійкості. В даному розділі, буде описано розробку стартап-проекту. В табл 4.1 викладено зміст ідеї, що пропонується, можливі напрямки застосування ідеї, основні вигоди, що може отримати користувач з ідеї та чим відрізняється ідея від існуючих аналогів.

Таблиця 4.1Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Мережа для хмарного сервісу на базі технології SDN	Побудова відмовостійкого сервісу хмарних послуг	Можливість для провайдера хмарних послуг запропонувати потенційному клієнту надійність свого сервісу
	Побудова незалежної від постачальників обладнання мережі	Можливість відмовитися від дорогого обладнання і прив’язки до обладнання конкретного постачальника

	Масштабованість мережі хмарного сервісу	Легка масштабованість мережі провайдера в разі різкої необхідності зміни топології мережі

Визначений перелік переваг ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності. Як видно з таблиці 4.1 перевагами є висока відмовостійкість, легка масштабованість та незалежність від вендорів при побудові хмарного сервісу.

4.2.Можливості запуску проекту

Таблиця 4.2 Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних конкурентів	>10
2	Динаміка ринку	Ринок технологій SDN стрімко набирає популярність, витісняючи традиційні мережеві рішення
3	Наявність та характер обмежень для входу	Відносно невелика кількість конкурентів компенсується наявністю великий компаній з серйозними фінансовими і дослідницькими ресурсами
4	Специфічні вимоги до стандартизації та сертифікації	За стандартизацію рішень в сфері SDN мереж відповідає організація ONF

Таблиця 4.3 Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в хмарних послугах	Корпоративні клієнти	Безпосередня взаємодія з кінцевим продуктом	Необхідність дешевих рішень
2	Стрімке поширення використання SDN	Провайдери хмарних послуг та	Розміри мереж	Підвищені вимоги до надійності і

		приватні компанії, що збираються переходити на SDN		масштабованості мережі

Таблиця 4.4Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Зависока конкуренція	Корпоративні клієнти Безпосередня взаємодія з кінцевим продуктом	Необхідність дешевих рішень
2	Обмежений функціонал	Провайдери хмарних послуг та приватні компанії, що збираються переходити на SDN Розміри мереж	Підвищені вимоги до надійності і масштабованості мережі

4.3. Технологічний аудит

Таблиця 4.5Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технології
1	Мережа для хмарного сервісу на базі технології SDN	NEC ProgrammableFlow Controller Software (PFC)	Є в наявності	Доступні по тріальній ліцензії
2		MiniNet OFV	Є в наявності	Доступні безкоштовно
3		Open vSwitch (OVS) 2.3	Є в наявності	Доступні безкоштовно
4		Вебслужба Apache Web Services	Є в наявності	Доступні безкоштовно
5		Реляційна система керування базами даних MariaDB	Є в наявності	Доступні безкоштовно
6		Система для організації хмарного сховища OwnCloud	Є в наявності	Доступні безкоштовно

7		Бібліотека PHP	Є в наявності	Доступні безкоштовно

Обрана технологія реалізації ідеї проекту: розробка мережі для хмарного сервісу на базі технології SDN з використанням мережевої ОС PFC, MiniNet OFV, OVS 2.3, вебслужби Apache Web Services, реляційної системи керування базами даних MariaDB, системи для організації хмарного сховища OwnCloud та бібліотеки PHP.

4.4. Розроблення ринкової стратегії продукту

Таблиця 4.6 Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Провайдери хмарних послуг	Не всі готові прийняти	Є в наявності	Висока	
2	Приватні компанії	Усі готові прийняти негайно	Є в наявності	Висока	
3	Державні установи	Не всі готові прийняти	Є в наявності	Низька	
4	Навчальні та дослідницькі програми	Не всі готові прийняти	Є в наявності	Середня	
З перерахованих цільових груп потрібно працювати з усіма, перевагу надавати приватним компаніям та державним установам					

Таблиця 4.7Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкуренто-спроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Демонстрація можливостей продукту шляхом публікації науково-популярних статей та відеоматеріалів	Співпраця з цільовими групами	Вільне розповсюдження для некомерційних користувачів	Оновлення продукту відповідно до побажань клієнтів

Таблиця 4.8 Визначення базової стратегії розвитку

№	Чи є проект першим в своєму роді на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде продукт копіювати основні характеристики конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Пошук нових користувачів	Основний функціонал буде схожим	Більш доступний сервіс при зберіганні функціоналу

4.5.Розроблення маркетингової стратегії стартап-проекту

Таблиця 4.9 Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Часте оновлення програми	Врахування побажань клієнтів та виправлення помилок в найкоротші терміни	Створення відділу підтримки для аналізування реагування на відгуки клієнтів
2	Формування звітів з результатами	Можливість формувати звіти по роботі	Модуль для формування звітів з результатами роботи програми

	моделювання	програми з детальним описом процесу моделювання та оцінкою ефективності мережі, що тестувалася	

Таблиця 4.10 Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Мережа для хмарного сервісу на базі технології SDN</td></tr>
<tr><td rowspan="7">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Модуль-редактор топології та пристроїв системи</td><td>250МБ</td></tr>
<tr><td>2. Модуль керування системою</td><td>50МБ</td></tr>
<tr><td>3. Модуль аналізу результатів та статистики</td><td>30МБ</td></tr>
<tr><td colspan="2">Якість: внутрішнє тестування програми, логування збоїв та система зворотнього зв’язку для повідомлення про неналежну роботу програми</td></tr>
<tr><td colspan="2">Пакування: продаж електронних ключів-ліцензій та інструкцією, що надсилається на електронну адресу замовника</td></tr>
<tr><td colspan="2">Марка: назва організації-розробника + назва товару</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу: реалізація системи</td></tr>
<tr><td colspan="2">Після продажу: електронна версія додатку з ключем</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання: прив’язка копії програмного продукту до конкретного ПК та активація програми шляхом введення ліцензійованого ключа, або шляхом надання послуг без передачі програмного продукту замовнику</td></tr>
</table>

Таблиця 4.11 Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1	0$ - 500$	0-500	>10000$/місяць	100-200

Таблиця 4.12 Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Покупка ліцензії на продукт або договір про надання послуг без передачі ПО до замовника	Розміщення на власному сайті для електронної дистрибуції	Канал збуту однорівневий (через роздрібну дистрибуцію)	Вертикальна (право власності залишається у розробника ПЗ)

Таблиця 4.13 Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Купують товар на вимогу	Тематичні зустрічі, конференції, презентації	Надання послуг тестування мереж	Демонстрація основних функцій розробленого	Охоплення аудиторії, пояснення функцій та

		тощо	розробленим продуктом, продаж ПЗ	продукту	можливо-стей розроблено-го продукту та його переваг

4.6. Висновки до розділу

Четвертий розділ присвячений розробці стартап-проекту для розроблюваного продукту, зроблено опис ідеї проекту, проведено детальний аналіз конкурентів та потенційних можливостей для реалізації продукту на даному ринку. Для дослідження в рамках розробки стартап-проекту було виконано наступні завдання:

1. Наведено загальний опис ідеї проекту для виходу на ринок, описаний приблизний функціонал проекту, визначено конкурентів та проведено порівняння функціоналу продуктів від конкурентів для визначення переваг та недоліків розроблюваного продукту в порівнянні з вищезгаданими конкурентами.
2. Проведено технічний аудит проекту і визначено можливості реалізації даного проекту.
3. Проаналізовано ринкові можливості для запуску проекту, наведено порівняльну характеристику перспектив запуску з конкурентами, створено план та ринкову стратегію розвитку продукту, визначено цільові групи та способи просування проекту в маси.
4. Розроблено маркетингову програму для просування продукту на ринку, описано способи та основні напрямки збуту, визначно першочергові цільові групи та маркетингові стратегії для розширення клієнтської бази.

В підсумку, було отримано стартап-проект для запуску розробленого програмного продукту на ринок, отримано навички створення стартап-проектів, побудови маркетингової стратегії та аналізу обраного ринку.

ЗАГАЛЬНІ ВИСНОВКИ

Дана магістерська робота присвячена дослідженню завдання побудови хмарного сервісу на основі програмно-конфігурованої мережі.

Було проведено огляд технології SDN з концептуальної, технологічної та ринкової точки зору, наведено її недоліки і переваги використання у реальних мережах, розглянуто надійність.

Окрім того було проведено аналіз сучасного стану хмарних технологій та описано перспективи застосування на основі мереж, створених базуючись на принципах SDN. Було переглянуто завдання побудови хмарного сервісу на основі програмно-конфігурованої мережі та наведено потенційний метод покращення її надійності в умовах корпоративного використання. Проведений аналіз існуючих рішень показав необхідність представлення нового підходу до виконання завдання побудови мережі на основі програмно-конфігурованих мереж через такі проблеми як неповне використання потенціалу технології SDN серед існуючих підходів яке призводить до зростання вірогідності відмови системи в разі виходу з ладу одного або кількох елементів складної корпоративної мережі.

Було досліджено і розглянуту широкий спектр підходів до забезпечення надійності та безперебійної роботи мережі, побудованої за принципами SDN.

З метою перевірки концепції алгоритму, створено тестовий стенд для моделювання роботи хмарного сховища на основі моделі програмно-конфігурованої мережі, у якому реалізація даного алгоритму покладена на контролер. Стенд дозволяє створювати, редагувати зберігати модель топології мережі, симулювати потоки трафіку вказаного розміру, числа пакетів і швидкості пересилання у мережі. Процес моделювання відбувається у режимі ручного керування користувача.

Аналіз результатів роботи виявив здатність забезпечувати достатній рівень надійності отриманої системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. H. Kim and N. Feamster, “Improving network management with software defined networking,” IEEE Communications Magazine, vol. 51, no. 2, pp. 114– 119, 2013.

2. D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.

3. H. Farhady, H. Lee, and A. Nakao, “Software-defined networking: A survey,” Computer Networks, vol. 81, pp. 79–95, 2015.

4. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus Networks,” SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

5. Open Networking Foundation, OpenFlow. www.opennetworking.org/sdn-resources/openflow. [Online; accessed January 2017].

6. ONOS Project, “Onos mission.” http://onosproject.org/mission/. [Online; accessed 20-September-2017].

7. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: towards an operating system for networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 13, pp. 105–110, 2008.

8. D. Erickson, “The Beacon OpenFlow Controller,” in Proceeding of the 2nd Workshop on Hot Topics in Software Defined Networking (HotSDN), pp. 13– 18, ACM, 2013.

9. D. Lenrow, “Intent As The Common Interface to Network Resources.”Presentation at the Intent Based Network Summit 2015, Palo Alto, CA, USA. Available at www.ietf.org/mail-archive/web/i2nsf/current/pdfEhAfL7kT9F.pdf(AccessedJanuary2017), 2015.

10. ONOS Project, “Intent-based framework.” http://wiki.onosproject.org/display/ONOS/Intent+Framework. [Online; accessed 20-September2017].

11. Болодурина И. П. Разработка методов и алгоритмов маршрутизации динамических потоков данных, приложений и сервисов в гетерогенной облачной платформе / И. П. Болодурина, Д. И. Парфёнов. // Фундаментальные исследования.. – 2012. – №12. – С. 24–30.

12. Смелянский Р.Л. Программно-конфигурируемые сети / Руслан Смелянский. // Открытые системы. СУБД. – 2012. – №9.

13. Dasari V. R. OpenFlow Arbitrated Programmable Network Channels for Managing Quantum Metadata [Електронний ресурс] / V. R. Dasari, T. S. Humble // The Journal of Defense Modeling & Simulation. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/288890479_OpenFlow_Arbitrated_Programmable_Network_Channels_for_Managing_Quantum_Metadata.

14. Моделювання роботи overlay мереж SDN та дослідження їх основних характеристик / Р. С.Одарченко, С. Ю. Даков, В. В. Поліщук, А. М. Тирсенко. // Наукоємні технології. – 2016. – С. 284–290.

15. Коляденко Ю. Ю. Организация программно-конфигурируемой сети на базе протокола OpenFlow / Ю. Ю. Коляденко, Е. Э. Белоусова. // Technology audit and prodoction reserves. – 2016.

16. Панеш А. Х. Достоинства и недостатки програмно-конфигурируемых компьютерных сетей / А. Х. Панеш. // Вестник АГУ. – 2016. – №3. – С. 109–113.

17. Ying - Dar Lin. OpenFlow Version Roadmap [Електронний ресурс] / Ying - Dar Lin. – 2015. – Режим доступу до ресурсу: http://speed.cis.nctu.edu.tw/~ydlin/miscpub/indep_frank.pdf.

18. McKeown N. OpenFlow: Enabling Innovation in Campus Networks [Електронний ресурс] / N. McKeown, T. Anderson, H. Balakrishnan // ACM

SIGCOMM Computer Communication Review. – 2008. – Режим доступу до ресурсу: http://ccr.sigcomm.org/online/files/p69-v38n2n-mckeown.pdf.

19. Лихачев В. А. Программно-конфигурируемые сети на основе протокола OpenFlow / В. А. Лихачев. // Приволжский научный вестник. – 2014. – С. 18–21. Обзор OpenFlow. [Електронний ресурс] // SDNBLOG. – 2015. – Режим доступу до ресурсу: https://sdnblog.ru/.

20. Альшаев И. А. Исследование принципов работы протокола OpenFlow в программно-конфигурируемых сетях / И. А. Альшаев, А. В. Красов, И. А. Ушаков. // Труды учебных заведений связи. – 2017. – С. 16–27.

21. What Is Google Espresso? [Електронний ресурс] // keycdn.com. – 2018. – Режим доступу до ресурсу: https://www.keycdn.com/support/google-espresso.

22. Vahdat A. Espresso makes Google cloud faster, more available and cost effective by extending SDN to the public internet [Електронний ресурс] / Amin Vahdat // blog.google.com. – 2017. – Режим доступу до ресурсу: https://www.blog.google/products/google-cloud/making-google-cloud-faster-more-available-and-cost-effective-extending-sdn-public-internet-espresso/.

23. Snover J. Software Defined Networking, Enabled in Windows Server 2012 and System Center 2012 SP1, Virtual Machine Manager [Електронний ресурс] / Jeffrey Snover // Windows Server Blog – Режим доступу до ресурсу: https://cloudblogs.microsoft.com/windowsserver/2012/08/22/software-defined-networking-enabled-in-windows-server-2012-and-system-center-2012-sp1-virtual-machine-manager/.

24. What's New in SDN for Windows Server 2019 [Електронний ресурс] // Microsoft Docs. – 10. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows-server/networking/sdn/sdn-whats-new.

25. SDN: альтернатива или дополнение к традиционным сетям? [Електронний ресурс] // Блог компании Hewlett Packard Enterprise. – 2015. – Режим доступу до ресурсу: https://habr.com/company/hpe/blog/255363/.

26. Кулаков Ю.О. Комп’ютерні мережі / Кулаков Ю.О., Луцький Г.М. // Підручник за редакцією Ю.С. Ковтанюка, – Київ.: Видавництво «Юніор», 2005. – 397с., іл.

27. Вишневский В. М. Теоретические основы проектирования компьютерных сетей / В. М. Вишневский. – Москва: Техносфера, 2003. – 512 с.

28. Пятибратов А. П. Вычислительные системы, сети и телекоммуникации / А. П. Пятибратов, Л. П. Гудыно, А. А. Кириченко. – Москва: Изд. центр ЕАОИ, 2009. – 292 с.

29. Ільченко М.Ю., Кравчук С.О. Телекомунікаційні системи. – Київ: Наукова думка, 2017. – 730 сКулаков Ю.А. Безопасная передача информации на основе мнопутевой маршрутизаци / Ю. А. Кулаков, А. О. Деревянчук. – 2009.

30. Досягнення в телекомунікаціях 2019 / за наук. ред. М.Ю.Ільченка, С.О.Кравчука: монографія. - Київ: Інститут обдарованої дитини НАПН України, 2019.- 336 с. Рекомендовано до друку ВР КПІ ім.І.Сікорського (прот.№10 від 04.11.2019 р.) ISBN

31. 978-617-7734-12-2Hung-Yun Hsieh. pTCP: An End-to-End Transport Layer Protocol for Striped Connections / Hung-Yun Hsieh, Raghupathy Sivakumar // IEEE Interna-tional Conference on Network Protocols (ICNP), Paris, France, November 2002, p. 1-10.

32. Keuntae Park. MTCP: A Transmission Control Protocol for Multi-Provider Environment / Keuntae Park, Yongin Choi, Donggook Kim, and Daeyeon Park // IEEE Consumer Communications and Networking Conference (CCNS), February 2006, p. 735-739.

33. Шуповалов В. П. КЛАССИФИКАЦИЯ МЕТОДОВ МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ / В. П. Шуповалов. // Т-СОММ: ТЕЛЕКОММУНИКАЦИИ И ТРАНСПОРТ. – 2014. – №1. – С. 29–32.

34. Еременко О. С. Потоковая модель многопутевой маршрутизации по непересекающимся путям в телекоммуникационной сети / О. С. Еременко. // Проблемы телекоммуникаций. – 2015. – №1. – С. 85–93.

35. Awduche D., Chiu A., Elwalid A., Widjaja I., Xiao X., «Overview and Principles of Internet Traffic Engineering», May 2002, IETF RFC 3272

36. Traffic Engineering in Software-Defined Networking: Measurement and Management / ZHAOGANG SHU, JIAFU WAN, JIAXIANG LIN та ін.]. // IEEE Access. – 2016. – №4. – С. 3246–3256.

37. Олифер В. Искусство оптимизации трафика / В. Олифер, Н. Олифер. // Журнал сетевых решений\LAN. – 2002. – №12.

38. Лассерр М. Использование VPLS делает операторские сети более гибкими / Марк Лассерр. // Журнал сетевых решений\LAN. – 2005.

39. Moy J. OSPF Version 2 [Електронний ресурс] / Moy // IETF RFC 2328 – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc2328.

40. Oran D. OSI IS-IS intra-domain routing protocol [Електронний ресурс] / Oran // IETF RFC 1142 – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc1142.