ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Створення системи віртуалізації мережі для покращення управління та забезпечення безпеки віртуальних мереж»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Михайло БІЛЯКОВ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КІД-42
Михайло БІЛЯКОВ
Ім'я, ПРІЗВИЩЕ
Керівник: Анастасія ВЄЧЕРКОВСЬКА
к.т.н., доцент, Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Білякову Михайлу Дмитровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Створення системи віртуалізації мережі для покращення управління та забезпечення безпеки віртуальних мереж
керівник кваліфікаційної роботи к.т.н., доцент Анастасія ВЄЧЕРКОВСЬКА,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «_____» _____2024р. № _____
2. Строк подання кваліфікаційної роботи «_____» _____ 2024р.
3. Вихідні дані до кваліфікаційної роботи:
   1. Утиліти аналізу віртуальних мереж
   2. інтернет ресурси стосовно віртуалізації мереж
   3. Науково-технічна література..
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Віртуалізація
2. Інтерфейси віртуальної мережі
3. Підвищення продуктивності та управління елементами віртуальної мережі
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «_____» _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	26.02-22.03.2024	Виконано
2	Обробка матеріалу	23.03-05.04.2024	Виконано
3	Розробка теоретичної частини Віртуалізація	06.04-10.04.2024	Виконано
4	Інтерфейси віртуальної мережі	11.04-30.04.2024	Виконано
5	Підвищення продуктивності та управління елементами віртуальної мережі	01.05-03.05.2024	Виконано
6	Висновки за результатами аналізу	04.05-08.05.2024	Виконано
7	Розробка презентації	09.05-11.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано
9	Здача в деканат	25.05-1.06.2024	Виконано

Здобувач(ка) вищої освіти _____ Михайло БІЛЯКОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Анастасія ВЄЧЕРКОВСЬКА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 73 стор., 32 рис., 20 джерел.

Мета роботи – Розробка та реалізація системи віртуалізації мережі, яка спрощує управління та забезпечує безпеку віртуальних мереж.

Об'єкт дослідження – процес віртуалізації мережі.

Предмет дослідження – віртуальні мережі.

Короткий зміст роботи: В дипломній роботі розглядається віртуалізація загалом, технології віртуалізації мереж, найпоширеніші інструменти віртуалізації та їхні позитивні й негативні показники, а також оцінюють ефективність застосування прототипів віртуалізації, використовуючи різні сценарії та зіставляючи результати показників утиліт на фізичному середовищі та віртуальному.

Також наведено аналіз інтерфейсів віртуальних мереж, їхні показники і тенденції, зокрема це внутрішні і зовнішні інтерфейси прототипів Xen і OpenFlow. Їхні відмінності та переваги.

У 3 розділі розглянуто елементи керування віртуальними мережами та методи підвищення ефективності роботи віртуальних мереж, зокрема прототипів Xen і OpenFlow.

КЛЮЧОВІ СЛОВА: ВІРТУАЛІЗАЦІЯ МЕРЕЖІ, УПРАВЛІННЯ МЕРЕЖЕЮ, БЕЗПЕКА МЕРЕЖІ, ВІРТУАЛЬНІ МЕРЕЖІ, XEN, OPENFLOW

ABSTRACT

The text part of the qualification work for the bachelor's degree: 73 pages, 32 figures, 20 sources.

The purpose of the work is Development and implementation of a network virtualization system that simplifies management and ensures the security of virtual networks.

The object of research is the process of network virtualization.

The subject of research is virtual networks.

Summary of the work: The thesis examines virtualization in general, network virtualization technologies, the most common virtualization tools and their positive and negative indicators, and evaluates the effectiveness of virtualization prototypes using various scenarios and comparing the results of utility indicators on the physical environment and the virtual environment.

The article also analyzes virtual network interfaces, their performance and trends, including internal and external interfaces of the Xen and OpenFlow prototypes, their differences and advantages.

Section 3 discusses virtual network controls and methods for improving the efficiency of virtual networks, including Xen and OpenFlow prototypes.

KEYWORDS: NETWORK VIRTUALIZATION, NETWORK MANAGEMENT, NETWORK SECURITY, VIRTUAL NETWORKS, XEN, OPENFLOW

ЗМІСТ

ВСТУП……...9
РОЗДІЛ 1. ВІРТУАЛІЗАЦІЯ …….11
1.1 Технології віртуалізації……..……..….…..11
1.2 Інструменти віртуалізації …….……..……13
1.3 Оцінювання ефективності ……..…....23
РОЗДІЛ 2. ІНТЕРФЕЙСИ ВІРТУАЛЬНОЇ …….…….....…30
2.1 Віртуальні мережі: продуктивність і тенденції ……..…….….30
2.1.1 Підходи до віртуалізації мережі……34
2.1.2 Технології мережевої віртуалізації. …….…....38
2.2 Xen прототип …….…….....44
2.3 OpenFlow прототип …….…….…49
РОЗДІЛ 3. ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ТА УПРАВЛІННЯ ЕЛЕМЕНТАМИ ВІРТУАЛЬНОЇ МЕРЕЖІ …….…….…55
3.1 Xen-based прототип……...55
3.2 OpenFlow прототип ……64
ВИСНОВКИ…….….……..72
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ…….…….…….74
Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)…….……76

ВСТУП

Специфіка сучасного ринку виробництва і послуг призводить до необхідності обробки величезних інформаційних потоків у реальному часі з підвищеними вимогами до безпеки та надійності. Для продуктивної роботи організацій потрібно все більше і більше сервісів, що надаються як для клієнтів, так і для власних співробітників. Запуск різноманітних служб і додатків на одному сервері веде до збільшення ризиків, пов'язаних з виходом його з ладу.

Для забезпечення мінімізації таких ризиків та ізоляції додатків один від одного очевидним видається рішення використовувати для кожного сервісу окремо виділений сервер. Такий підхід призводить до швидкого зростання мереж та інженерних комунікацій.

Віртуалізація - це абстраговані від фізичних (апаратних) апаратної реалізації, комплекси обчислювальних ресурсів, які повністю ізольовані між собою, перебуваючи при цьому в одній фізичній машині. Віртуальні мережі, своєю чергою, виступають тут у ролі віртуальних комунікаційних ресурсів. Широко використовуються дві реалізації цієї технології: VPN (Virtual Private Network) і VLAN (Virtual Local Area Network).

VPN - віртуальна приватна мережа, це логічна мережа (одне або більше з'єднань), реалізована за верхом іншої мережі (мережі Інтернет зокрема), використовуючи шифрування. Цю технологію часто використовують для з'єднання кількох віддалених локальних мереж. Як приклад можна показати комунікація офісом географічно віддалених один від одного або ж CYOD (Choose Your Own Device) чи BYOD (Bring Your Own Device).

VLAN - віртуальна локальна мережа, своєю чергою, є логічною мережею, розділеною на підмережі фізичної мережі. Як приклад можна уявити поділ фізичної мережі будівлі на логічні мережі відділів. Ці мережі можуть бути повністю ізольовані і працювати за заздалегідь налаштованими правилами передачі даних. Ці технології повсюдно використовуються скрізь, але існують

кілька технологій реалізації та різні підрядники цих технологій. У даній роботі розглядається саме прототипи Xen і OpenFlow як два найпоширеніших. Також розглядаються продуктивність, показники, інтерфейси, можливості тощо.

Актуальність цієї теми полягає в побудові системи захисту даних за допомогою віртуалізації з шифруванням інформації для захисту конфіденційності та цілісності даних зі збереженням їхньої доступності.

РОЗДІЛ 1. ВІРТУАЛІЗАЦІЯ

1.1 Технології віртуалізації

Для кращого розуміння засобів віртуалізації, важливо визначити відмінності між технологіями віртуалізації. У цьому розділі кваліфікаційної роботи описано концепції та методи, що використовуються Xen, VMware і OpenVZ.

Існує безліч різних визначень віртуалізації, але всі вони схожі в тому, що ці технології існують для спільного використання обчислювальних ресурсів і надання певного рівня ізоляції між віртуальними середовищами. Класичним визначенням, згідно з Попеком і Голдбергом, є те, що віртуалізація – це технологія, яка забезпечує віртуальні середовища, що є ефективними та ізольованими подібно до фізичного обладнання.

Сьогодні, це поняття може бути узагальнене не тільки на апаратних засобах, а й на будь-якому обчислювальному ресурсі, такому як ядро ОС або абстракція VM, при використанні мов програмування Java, C# тощо.

При цьому в результаті віртуалізації виникають кілька проблем. Перша проблема, що стосується всіх віртуальних середовищ, - це яким чином розділені віртуальні образи отримують доступ до основних обчислювальних ресурсів. Для віртуалізації апаратних засобів існують загальні ресурси - це центральний процесор, RAM, довгострокова пам'ять і мережа. Поділ по відношенню до RAM-у може бути реалізовано по-різному. Віртуальному середовищу можна надати доступ до простору віртуальної пам'яті, який має посилання на блоки пам'яті фізичного RAM-а, тим самим способом, яким користується ОС. Інший підхід повинен дати VMs можливість отримати знання про тип віртуалізації і безпосередньо отримувати доступ до пам'яті, виділеної гіпервізору.

Центральний процесор може бути представлений кількома способами і здійснюватися, використовуючи такі механізми, як кругова система, зважена кругова система, розподіл за вимогами тощо.

Введення/виведення, загалом, може бути оброблено загальним способом, використовуючи буфери для зберігання переданих даних, які є мультиплексними і не мультиплексними між фізичною і віртуальною периферією. Сьогодні на практиці використовують продукти з архітектурою x86, які спричиняють додаткові проблеми віртуалізації. На початку розвитку віртуалізації, а саме в 1970-х, апаратні засоби були розроблені для використання у віртуалізації. У цій архітектурі центрального процесора віртуалізація апаратних засобів могла бути досягнута за допомогою використання техніки, названої «de-privileging». За словами Попека і Голдберга, є три вимоги для реалізації гіпервізора над фізичними апаратними засобами:

- ефективність, що означає, що численна підгрупа - це набір команд від віртуального центрального процесора (vCPU), які мають бути виконані безпосередньо у реальному центральному процесорі без будь-якого виду втручання гіпервізора;

- контроль за ресурсом, що означає, що гіпервізор повинен мати повний контроль над усіма ресурсами;

- еквівалентність, що означає, що гіпервізор повинен надати віртуальний інтерфейс віртуальному середовищу, еквівалентного оригінальному інтерфейсу.

Повна віртуалізація. Повна віртуалізація - метод віртуалізації, у якому віртуальні всі оригінальні інтерфейси, причому, інтерфейси, виставлені у віртуальному середовищі, ідентичні оригінальному інтерфейсу. В цьому підході гостьова ОС, що знаходиться у VM, не повинна бути змінена і безпосередньо виконується у VM.

Спільні зусилля найбільших виробників апаратних засобів спрямовані на оптимізацію процесу віртуалізації. Консолідація сервера використовує віртуалізацію для заміни кількох серверів одним єдиним апаратним засобом із вищими показниками, де виконується робота кількох віртуальних серверів. Консолідація сервера стала звичайною практикою, спрямованою на скорочення обладнання та витрат на обслуговування в найбільших компаніях. Тому і AMD, і Intel розробили технології для більш ефективної підтримки віртуалізації в

сучасних центральних процесорах. Intel Virtualization Technology (IVT) і віртуалізація AMD (AMD-V) забезпечили кращу роботу для повної віртуалізації, запровадивши два нові режими функціонування: кореневий і некореневий. Режим функціонування кореня використовується гіпервізором і подібний до регулярної операції в центральному процесорі, забезпечуючи повний контроль за центральним процесором і реалізуючи традиційний метод чотирьох рівнів привілеїв. Некореневий спосіб призначається для виконання з боку VM-ів. У цьому способі центральний процесор також ініціює чотири рівні привілеїв і гостя, у якому ОС більше не припиняє процес на рівні «de-privileged», а здійснює цю дію на початковому рівні. Щоразу, коли гостьова ОС виконує проблематичну інструкцію, центральний процесор перехоплює її і повертає контроль до гіпервізора, щоб використовувати цю інструкцію.

Паравіртуалізація. Паравіртуалізація - це метод віртуалізації, у якому якому гостьова ОС співпрацює з гіпервізором, для здійснення кращої роботи. У паравіртуалізації гостем називається та ОС, яку змінено, коли виконується проблематична інструкція. Чутлива інструкція замінена обізнаною про віртуалізацію інструкцією і називається гіпервізором. Тому гіпервізор не повинен контролювати виконання VM для проблематичних інструкцій. Компромісом є те, що ОС повинна бути змінена і повторно запущена, щоб виконати паравіртуалізовані образи ОС. Паравіртуалізований образ ОС необхідний, щоб робити виклики гіпервізора. Це перешкоджає віртуалізації застарілої ОС і вимагає співпраці з розробниками ОС.

1.2 Інструменти віртуалізації

У цьому розділі дипломної роботи описуються головні методи віртуалізації, і будуть представлені результати оцінки діяльності для Xen, VMware і OpenVZ.

Xen - загальнодоступний гіпервізор, запропонований для роботи на товарних платформах апаратних засобів, які використовують метод

паравіртуалізації. Xen дає нам змогу одночасно керувати багатофункціональними VMs на єдиній фізичній машині.

Архітектура Xen складається з одного гіпервізора, розташованого вище за фізичних апаратних засобів і декількох VMs по гіпервізору. У кожного VM може бути своя власна ОС і додатки. Гіпервізор керує доступом до апаратних засобів, а також наявними ресурсами, розділеними між VMs. Крім того, драйвери пристроїв збережені в ізольованому VM, названому Доменом 0 (dom0) для забезпечення надійної та ефективної апаратної підтримки. Оскільки dom0 має повний доступ до апаратних засобів фізичної машини, у нього існують також спеціальні привілеї порівняно з іншими VMs, які називаються Користувацькими Доменами (domUs). З іншого боку, domUs мають віртуальні драйвери, названі драйверами фронтенда, які спілкуються з драйверами бекенда, розташованими в dom0 для отримання доступу до фізичних апаратних засобів.

Далі, коротко представлено пояснення, як Xen віртуалізує кожен машинний ресурс для віртуального маршрутизатора: процесор, пам'ять і ввід/вивід пристрої.

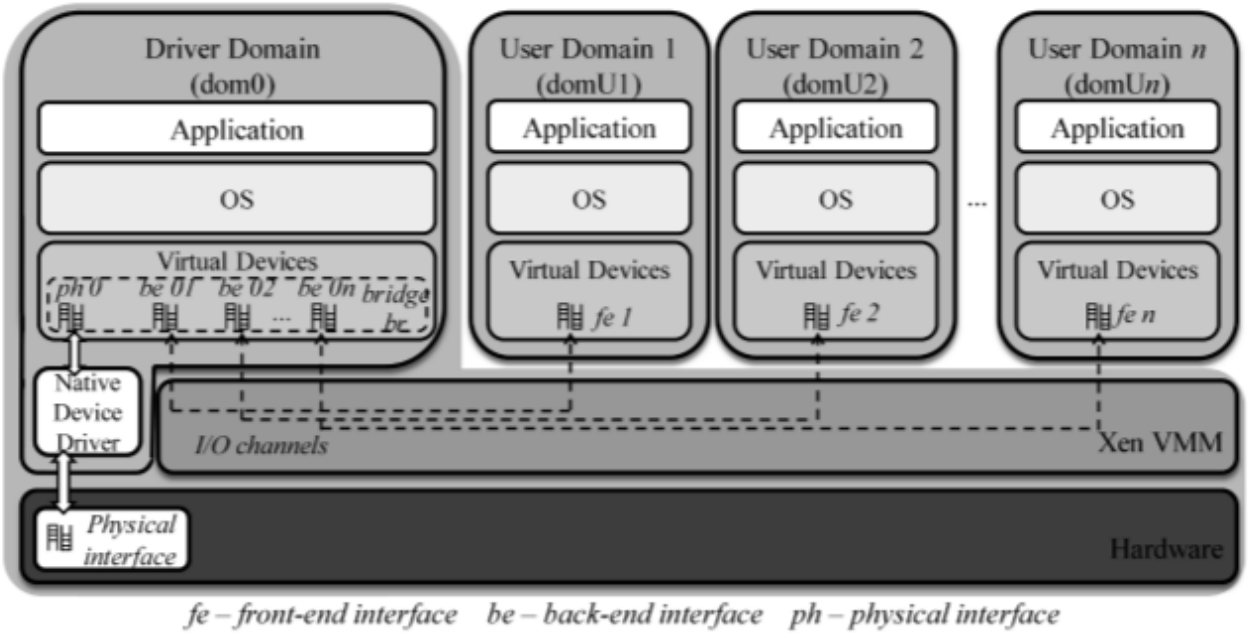


Рисунок 1.1 - Xen архітектура

Xen віртуалізує процесор, призначаючи vCPUs до VMs. vCPUs – це центральні процесори, які бачать процеси, що йдуть у кожній VM.

Гіпервізор Xen є планувальником центрального процесора, який динамічно розподіляє процесорний час фізичного центрального процесора між кожним vCPU під час певного періоду.

Планувальник за замовчуванням використовуваної Xen - планувальник кредиту. Планувальник кредиту перерозподіляє ресурси центрального процесора кожному VM (або, більш точно, кожному vCPU) згідно з параметрами, призначеними на VMs. Планувальник кредиту може також бути збережений для роботи на Symmetric Multi Processing (SMP). Це означає, що планувальник дозволяє фізичним центральним процесорам досягати 100%, якщо який-небудь VM повинен виконати роботу. У планувальнику, що зберігає роботу планувальнику немає жодного обмеження на суму ресурсів центрального процесора, які може використовувати VM. Розподіл пам'яті в Xen в даний час робиться статично. Кожен VM отримує встановлений обсяг пам'яті, який визначено під час його створення. На додаток, щоб вимагати мінімальної участі від гіперщитка, VMs відповідальні за розподіл і управління відповідною частиною таблиць сторінок апаратних засобів. Щоразу, коли VM вимагає нові таблиці сторінок, він асигнує та ініціалізує сторінку від свого власного місця в пам'яті та реєструє її в гіперщитку Xen, який відповідальний за забезпечення ізоляції. У Xen дані з пристроїв введення/виведення передані до кожного і від кожного VM, що використовує спільно пам'ять асинхронного буферного дескрипторного рівня. Завдання гіперщитка Xen полягає в тому, щоб виконати ратифікації перевірки. Пристрої введення/виводу доступні Dom0 безпосередньо за допомогою його рідних драйверів і також виконують операції з введення/виведення від імені domUs. З іншого боку, domUs використовують свої драйвери бекенда, щоб запросити доступ пристрою від dom0.

Особливий випадок віртуалізації вводу/виводу - мережева віртуалізація, яка відповідальна за демультиплексування пакетів, що надходять від фізичних інтерфейсів до VMs і також для мультиплексування комунікаційних пакетів, вироблених VMs. Рисунок 1.2 ілюструє використовувану Xen архітектуру мережі за замовчуванням. Для кожного domU Xen створює інтерфейси віртуальної

мережі, необхідні цим domU. Ці інтерфейси називаються інтерфейсами фронтенда і використовуються domUs для усіх його мережевих комунікацій. Крім того, внутрішні інтерфейси створюються в dom0 і відповідають кожному інтерфейсу фронтенда в domU.

Щоб обмінюватися даними між бекендом і інтерфейсами фронтенда, Xen використовує канал вводу/виводу, який, своєю чергою, використовує механізм нульової копії, в якому внутрішні інтерфейси вступають в ролі повноважень для віртуальних інтерфейсів у dom0. Фронтенд і внутрішні інтерфейси пов'язані один з одним через канал введення/виведення. На рисунку 1.2 внутрішні інтерфейси в dom0 пов'язані з фізичми інтерфейсами, а також один з одним через міст віртуальної мережі. Цю архітектуру за замовчуванням, використовувану Xen, називають з'єднаним способом. Таким чином і канал введення/виводу, і мережевий міст встановлюють канал зв'язку між віртуальними інтерфейсами, створеними в domUs через фізичний інтерфейс.

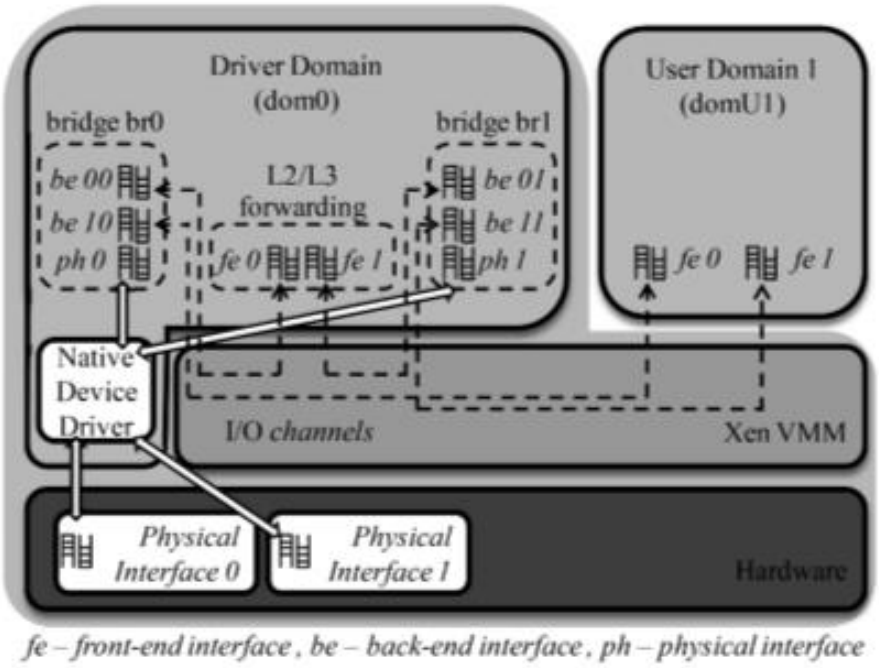


Рисунок 1.2 - Архітектура мережі Xen

VMware - компанія, яка надає машинні платформи віртуалізації клієнтам центру обробки даних віртуалізації клієнтам центру обробки даних і кінцевому користувачеві. Платформи віртуалізації VMware засновані на понятті повної

віртуалізації. Воно оцінює рівень платформи віртуалізації центру обробки даних VMware під назвою VMware Сервер ESX.

Ізоляція VM і частота представлення ресурсу на основі політики розподілу ресурсів встановлені системним адміністратором.

Ресурс, що розділяє, динамічний, оскільки ресурси можуть бути визначені та перерозподілені до VMs на вимогу. Архітектура VMware, як показано на рисунку 1.3 складається з компонентів інтерфейсу апаратних засобів, монітора віртуальної машини (VMM), VMkernel, менеджера ресурсів і сервісного управління. Компоненти інтерфейсу апаратних засобів відповідальні за здійснення визначених для апаратних засобів функцій і створюють надану VMs абстракцію апаратних засобів.

Це робить незалежні апаратні засоби VM як VMM відповідальними за центральний процесор віртуалізації, надаючи vCPU кожному VM. VMkernel керує і стежить за основними апаратними засобами. VMM і VMkernel разом здійснюють шар віртуалізації. Управління ресурсами здійснюється VMkernel. Він ділить основні фізичні ресурси між VM, перерозподіляючи ресурси для кожного VM. Сервісна панель надає безліч послуг, таких як самоналаштування, ініціювання виконання шару віртуалізації, управління ресурсами, запуск додатку, що здійснює підтримку управління.

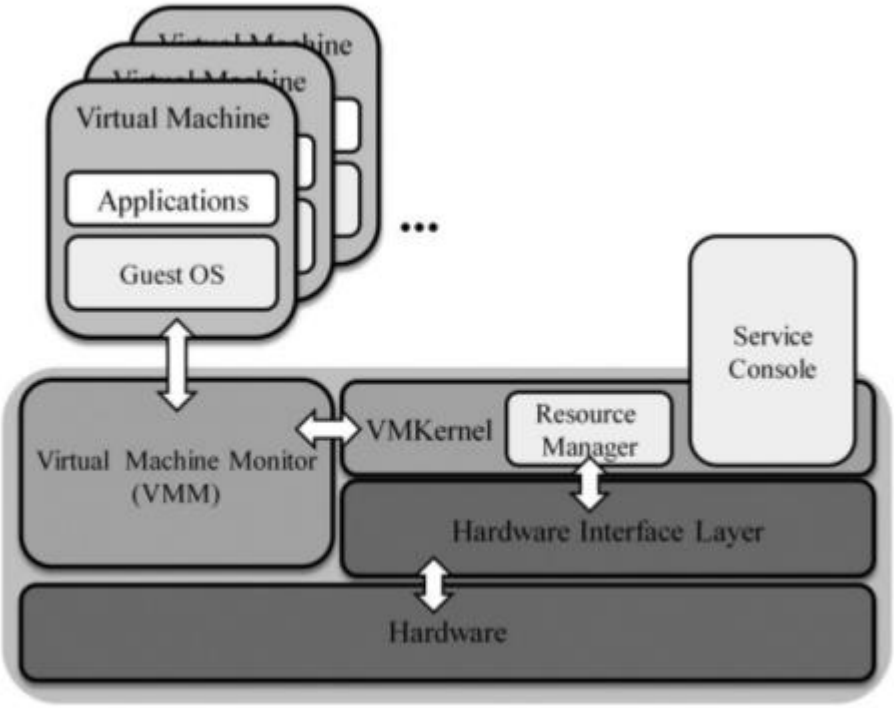


Рисунок 1.3 - Архітектура VMware

VMware Сервер ESX, як і інші інструменти віртуалізації, віртуалізує чотири головні ресурси: центральний процесор, пам'ять, диск і мережевий пристрій. Віртуалізація центрального процесора здійснюється за допомогою встановлення vCPU для кожного VM. VM не розуміє, що запускається через vCPU, тому що у vCPUs, повинні бути свої власні регістри і структури контролю. У VM можуть бути один або два vCPUs. Коли у нього є більше, ніж один центральний процесор, він називається SMP VM. VMM відповідальний за віртуалізацію центрального процесора, встановлює етапи реалізації та виконує інструкції. В віртуалізованому навколишньому середовищі гостьова ОС перебуває на більш нижчому привілейованому рівні, ніж той, на якому її було запущено.

VMware об'єднує два методи віртуалізації центрального процесора: пряме виконання та емуляція центрального процесора. Інструкції від простору користувача VM виконані безпосередньо на фізичному центральному процесорі технологією, відомою як пряме виконання.

Інструкції, які чутливі до рівня привілею, відбираються VMM, який слідує до прикладу виконання інструкції, представляючи початковий процес. Об'єднання

обох методів надає центральному процесору інтенсивний простір. Втрата виконавчих показників залежить від кількості точних інструкцій, які мали бути замінені. Планування центрального процесора виконується менеджером ресурсів і засноване на діях, що надають дані про те, скільки часу дано кожному VM.

Планування центрального процесора - пропорційна дія, що означає, що час центрального процесора, даний кожному VM, пропорційний. У SMP VM розподіл центрального процесора відрізняється. Менеджер ресурсів призначає vCPUs безпосередньо на фізичні центральні процесори і намагається виконати їх у той самий час.

Планувальник центрального процесора VMware намагається тримати рівновагу між розподілом центрального процесора VM. Коли VM буде зупинено або перестане працювати, менеджер ресурсів призначає його час використання центральним процесором до інших VMs. Підхід віртуалізації пам'яті полягає в тому, що VMware має створити новий рівень переведення адреси пам'яті. Це робиться шляхом надання кожній гостьовій ОС віртуальної таблиці сторінок за допомогою Memory Management

Unit (MMU). У середовищі віртуалізації VMware гостьова ОС отримує доступ до простору віртуальної пам'яті, наданої VM. Гостьова таблиця сторінок ОС підтримує послідовність між гостьовою віртуальною сторінкою та гостьовою віртуальною «фізичною» сторінкою. Гостьова віртуальна сторінка є сторінкою віртуальної пам'яті як у VM, так і в рідній віртуальній пам'яті ОС. Однак, гість віртуального механізму не може отримати доступ безпосередньо до фізичної пам'яті, він отримує доступ до гостьової віртуальної «фізичної» пам'яті. Гостьова віртуальна «фізична» пам'ять «фізична» пам'ять є абстракцією фізичної пам'яті. Коли гостьова ОС намагається виконати інструкцію й отримати доступ до фізичної пам'яті, виконується інструкція VMM, і його адреса переводиться за реальною фізичною адресою. Роздільна пам'ять VMware підкоряється політиці адміністрації, яка, наприклад, визначає мінімальну і максимальну суму фізичної пам'яті, до якої отримає доступ VM.

Також можна мати VMs, які споживають більше, ніж загальна сума фізичної пам'яті, доступної на фізичній машині. Це можливо, тому що що хост-система може також здійснювати обмін, як і в традиційному механізмі віртуальної пам'яті, використовуваному в сучасній ОС. Підхід віртуалізації вводу/виводу VMware має наслідувати критичні за відношенню до роботи пристроїв, таких як диск і мережеві карти. Доступні пристрої, емульовані VMkernel. VMkernel - так називається шар інтерфейсу апаратних засобів, який відповідальний за доступ до драйвера пристрою та виконання операції на фізичному пристрої. Для віртуалізації зберігання у VM представлений драйвер Small Computer System Interface (SCSI). Щодо віртуалізації мережі I/O VMware здійснює vmxnet драйвер пристрою, який є абстракцією основного фізичного пристрою. Коли додаток хоче надіслати дані через мережу, гостьова ОС обробляє запит і викликає vmxnet драйвер пристрою.

Запит введення/виведення отримує VMM, і контроль передається VMkernel. VMkernel незалежний від фізичного пристрою. Він обробляє запит, справляється з різними запитами VM і викликає рівень інтерфейсу інтерфейсу апаратних засобів, які здійснюються певним драйвером пристрою. VMware Сервер ESX збирає групи з відправлення або отримання мережевих пакетів перш, ніж зробити передачу контексту. Цей механізм використовується тільки тоді, коли пріоритет пакета досить високий, щоб не збільшувати затримки пакета. На закінчення, Сервер ESX VMware - інструмент віртуалізації, який надає широке управління та адміністрування. VMware Сервер ESX зосереджений на віртуалізації центру обробки даних. Він забезпечує гнучку і високоефективну центральну віртуалізацію пам'яті. Однак введення/виведення віртуалізації все ще залишається проблемною, оскільки вона відбувається на основі фізичних пристроїв і включає зміни контексту.

OpenVZ - загальнодоступний інструмент віртуалізації рівня ОС. OpenVZ дозволяє багаторазове ізольоване навколишнє середовище виконання за єдиним ядром ОС. Кожне ізольоване навколишнє середовище виконання називають Virtual Private Server (VPS). VPS схожий на фізичний сервер, маючи його власні

процеси, користувачів, файли, адреси Internet Protocol (IP), системну конфігурацію і забезпечуючи повний доступ до кореня. Головне місце використання цієї технології віртуалізації є веб-хостинг, де надається кожному клієнту повне навколишнє середовище Linux, а так само вона використовується в освітніх установах інформаційних технологій (IT), надаючи кожному студенту сервер Linux, який може перевірятися і управлятися віддалено.

OpenVZ менш гнучкий, ніж інші інструменти віртуалізації, такі як VMware або Xen, тому що навколишнє середовище для OpenVZ має бути Linux дистрибутивом, на основі того ж самого ядра ОС фізичного сервера.

Архітектура OpenVZ, як показано на рисунку 1.4 складається зі зміненого ядра Linux, яке знаходиться вище апаратних засобів. OpenVZ - змінене ядро, що здійснює віртуалізацію та ізоляцію декількох підсистем, управління ресурсом і контрольно-пропускними пунктами. Крім того, механізми віртуалізації вводу/виводу забезпечені OpenVZ-зміненим ядром, у якого є драйвер пристрою для кожного пристрою вводу/виводу. Це змінене ядро також здійснює дворівневий планувальник процесу, який відповідальний за перший рівень, визначаючи який VPS буде виконуватися, і за рішення хто буде керувати процесами VPS. Дворівневий планувальник і деякі особливості, які забезпечують ізоляцію між VPSs, формують шар віртуалізації OpenVZ. VPS-и виконуються вище шару віртуалізації

OpenVZ. У кожного VPS є свій власний набір заяв і пакетів, які є сегментаціями певних розподілів Linux і містять заяви або послуги. Тому у VPS можуть бути свої власні послуги та заяви, незалежні одна від одної.

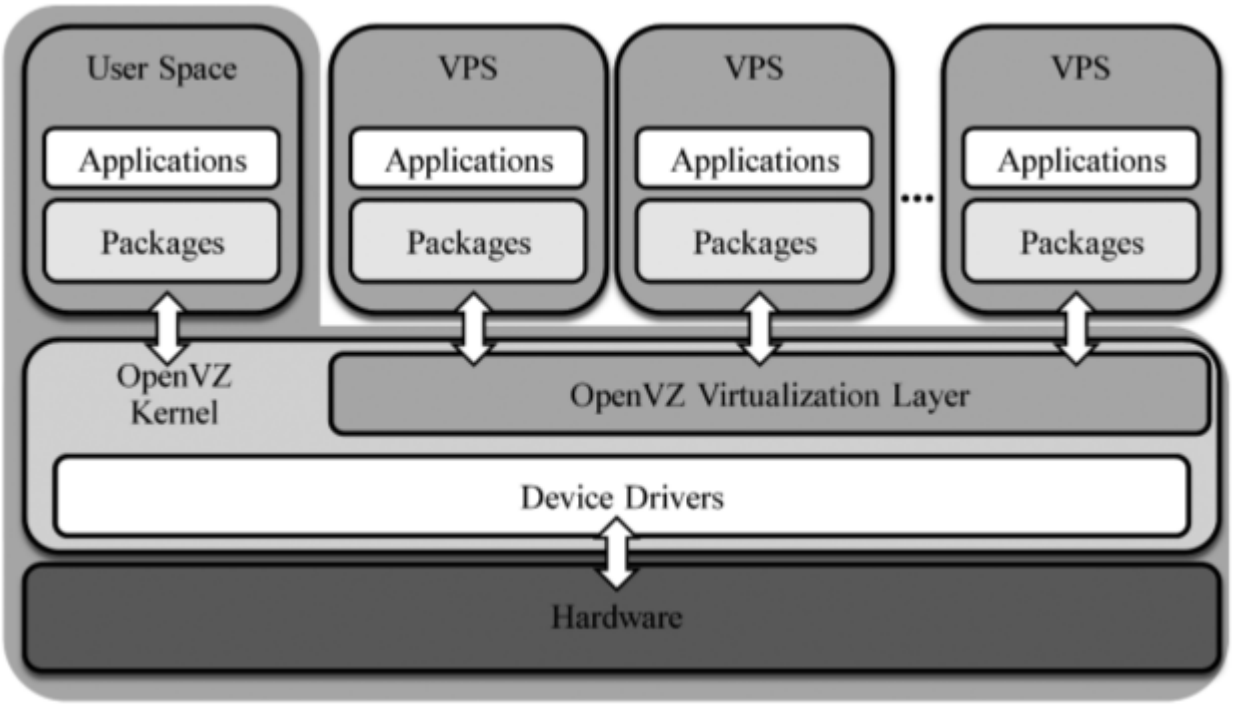


Рисунок 1.4 - Архітектура OpenVZ

Віртуалізація ресурсу в OpenVZ вирішує, дозволивши або заборонивши, що VPS отримає доступ до ресурсу на фізичному сервері. Загалом ресурси в OpenVZ не емульовані, вони розділені між VPS-ми. Визначення значення кожного ресурсу, який гарантується для кожного VPS, знаходиться в конфігураційному файлі VPS. Для віртуалізації процесора OpenVZ здійснює дворівневий планувальник центрального процесора. На першому рівні вирішує шар віртуалізації, який VPS виконає протягом кожного інтервалу часу, беручи до уваги пріоритет центрального процесора VPS, виміряний в одиницях CPU. На другому рівні, який виконується у VPS, стандартний планувальник Linux визначає, який процес виконає протягом кожного інтервалу часу, беручи до уваги стандартні пріоритетні параметри процесу.

OpenVZ дозволяє VPS безпосередньо отримувати доступ до пам'яті. Крім того, це більш гнучко, ніж інші технології віртуалізації. Під час виконання VPS сума пам'яті, присвячена одному VPS, може бути динамічно змінена адміністратором. Ядро OpenVZ керує місцем у пам'яті VPS, щоб тримати у фізичній пам'яті блок віртуальної пам'яті, що відповідає управлінню VPS.

OpenVZ віртуальний диск є поділом файлової системи фізичної машини. Так само до планування центрального процесора, дискове використання OpenVZ визначено дворівневою дисковою квотою.

На першому рівні шар віртуалізації OpenVZ визначає дискову квоту на кожен VPS, наприклад, обмежуючи максимальний розмір папки у файловій системі. На другому рівні можливо визначити дискові квоти на користувачів і групи у VPS, використовуючи стандартні механізми квоти Linux. Мережевий шар віртуалізації ізолює VPS-и один від одного і від фізичної мережі. Механізм віртуалізації мережі за замовчуванням OpenVZ створює інтерфейс віртуальної мережі для VPS і призначає IP-адресу на нього в хост-системі. Коли пакет прибуває в хост-систему з IP-адресою призначення VPS, маршрути хост-системи пакета до відповідного VPS. Цей підхід мережевої віртуалізації дає змогу пакетам VPS бути отриманими та надісланими, використовуючи модуль направлення хост-системи. Це спрощує мережеву віртуалізацію, але вводить додатковий переклад у пакетах маршруту.

1.3 Оцінювання ефективності

Оцінювання результатів діяльності. У цій роботі наведено результат експериментів із VM, який порівнює роботу, отриману з Xen, VMware і OpenVZ проти рідного Linux. Було використано певні оцінки для центрального процесора, RAM, жорсткого диска і мережевих ресурсів.

Продуктивність ЦП Тест Суперпі - інтенсивне центральним процесором завдання на основі алгоритму Гаусса-Лежандра, щоб обчислити вартість Пі. Алгоритм Гаусса-Лежандра повторюваний і заснований на багатьох арифметичних операціях. Для цього тесту скрипт оболонки обчислює вартість Пі з 222 цифрами (4,194,304 цифри) для десяти раундів. Виконавча метрика - час, витрачений, щоб обчислити Пі в секундах.

Робота пам'яті. Розподіл пам'яті, Набір і Рід (MASR) є інструментом порівняльного аналізу пам'яті, розробленим Teleinformatic and Automation Group

(GTA) з Федерального університету Ріо-де-Жанейро. MASR визначає ефективність пам'яті, призначаючи 2 ГБ пам'яті гіпервізору, послідовно встановлюючи всі значення пам'яті в постійне значення і послідовно читаючи всі дані з пам'яті. MASR був розвинений для порівняльного аналізу пам'яті з детермінованою кількістю операцій, незалежних від роботи комп'ютера. Оскільки жодні інструменти порівняльного аналізу пам'яті Linux не було знайдено з цією явною особливістю, було створено MASR. Для цього тесту було розроблено скрипт оболонки, в якому MASR виконується за 10 кроків. Оцінений параметр вимагає час, щоб виконати критерій MASR у кожному раунді.

Бонні ++. Бонні ++ є загальнодоступною дисковою утилітою, розробленою, щоб оцінити роботу файлової системи та жорсткий диск.

Головна програма Бонні ++ перевіряє у 20-ти віртуальних мережах один тимчасовий файл в 1 ГБ або багато тимчасових файлів загальним обсягом в 1 ГБ для великих об'ємів даних, з доступом до бази даних, моделюючи операції такі, як створення, читання та видалення великих і маленьких тимчасових файлів. Друга програма перевіряє виконання різних областей жорсткого диска, читаючи дані з блоків, розташованих на початку, в середині і в секторах кінця жорсткого диска. Для цього тесту скрипт оболонки призупиняє Бонні ++ за 10 раундів із параметром для використання 2 ГБ дискового простору. Виконавча метрика - час, витрачений, щоб призупинити Бонні ++, визначаючи щоразу ефективність.

ZFG Zero File Generator. ZFG є дисковим інструментом порівняльного аналізу, розробленим GTA/UFRJ. ZFG визначає ефективність диска безперервною швидкістю запису, створюючи бінарний файл на 2 ГБ, заповнений нулями 10 разів у кожному раунді. ZFG був розроблено для порівняльного аналізу диска з детермінованою кількістю операцій, незалежний від роботи комп'ютера, оскільки з такою явною ознакою жодні дискові інструменти порівняльного аналізу Linux не було знайдено. Для цього тесту було розвинуто скрипт оболонки, в якому ZFG виконано за 10 раундів. Оцінений параметр - час, витрачений, щоб виконати оцінку ZFG на кожному раунді.

Продуктивність мережі. Iperf - загальнодоступна мережева оцінка, яка дає змогу визначати ефективність мереж, використовуючи і односпрямовані та двоспрямовані потоки даних за протоколом Transmission Control Protocol (TCP) або User Datagram Protocol (UDP). У Iperf є кілька параметрів, які можуть формуватися, такі як розмір пакета, пропускна здатність тощо. У цьому тесті використано Iperf, сформований з односпрямованими потоками даних UDP, використовуючи 1,472 байта корисного вантажу, щоб уникнути IP фрагментації. Для цього тесту було розроблено скрипт оболонки, у якому односпрямований потік даних UDP йде або з випробувального комп'ютера на комп'ютер тестування мережі, або у протилежному напрямку.

Суперпі. Результати для оцінки Суперпі, виконаної за 10 циклів, показують на рисунку 1.5. У середньому час виконання на раунду тесту показує вертикальна вісь, найменший показник найкращий.

Інструменти віртуалізації та користувацькі Linux розподілені вздовж горизонтальної осі.

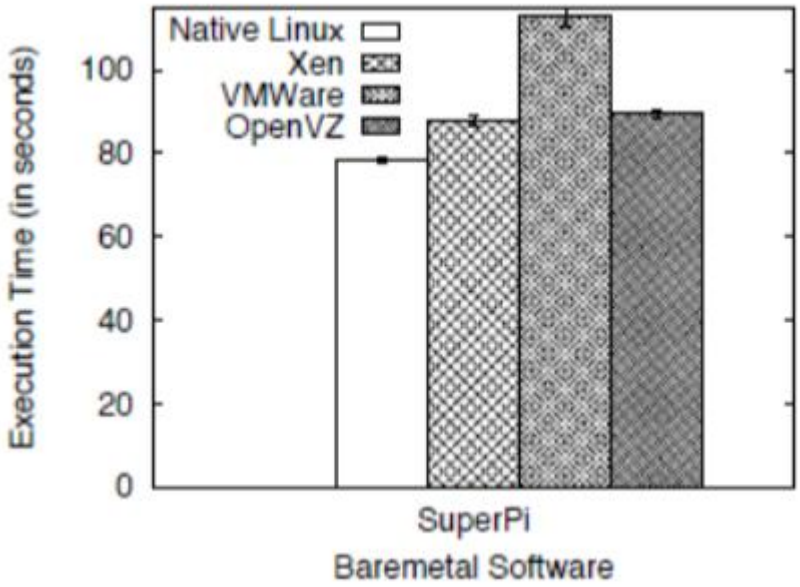


Рисунок 1.5 - Тест Суперпі

Як очікувалося, невіртуальна система виграє у віртуальної системи. Це було очікувано, оскільки паравіртуалізований VM Xen може безпосередньо отримати доступ до пам'яті після схвалення гіперщитка для використання області пам'яті. У

VMware найгірший показник, наслідок додаткового переведення адреси між віртуальною передбачуваною RAM до VM і фізичної RAM, і також до планувальника центрального процесора VM. У OpenVZ трохи вищі показники, ніж Xen, через механізм поділу планування між контейнерами центрального процесора.

MASR. Результати для оцінки MASR, виконаної за 10 циклів, показано на рисунку 1.6. Середній час виконання для циклу тесту показано на вертикальній осі, найменші показники кращі. Інструменти віртуалізації та Linux поширені вздовж горизонтальної осі.

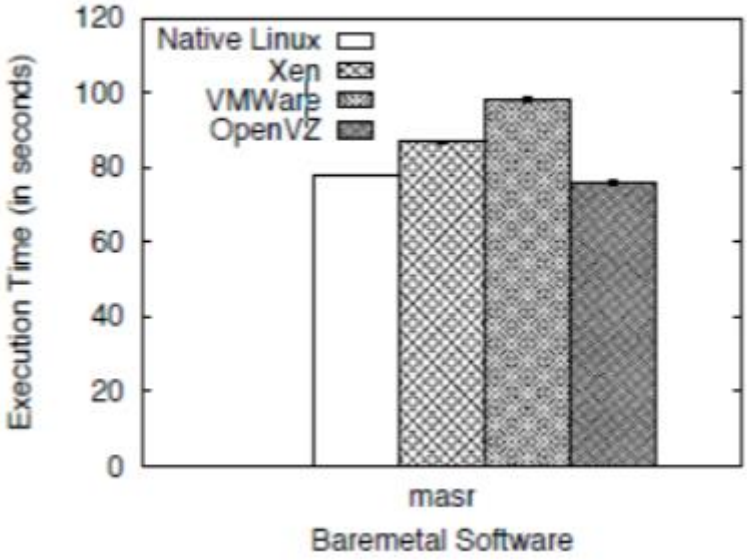


Рисунок 1.6 - Тест MASR

Результати показують, що у Linux, розгорнутого над фізичною оболонкою, найкращі показники в роботі, як і очікувалося. OpenVZ досягає подібними показниками роботи до Linux, оскільки OpenVZ ділить пам'ять динамічно з усіма контейнерами з низьким рівнем ізоляції, який лише гарантує мінімальну суму приватної пам'яті для кожного контейнера. Показники Xen трохи кращі, ніж VMware. VMware гірше працює через додаткове переведення адреси пам'яті та планувальника VM.

Усі інструменти віртуалізації представили прийнятний пороговий показник, який є дуже важливим результатом, оскільки операції з доступу пам'яті дуже поширені у відправленні пошуку за таблицею та інших завдань напряму.

Бонні ++. Результати для Бонні ++ дискова оцінка, виконана за 10циклів, показані на рисунку 1.7. Середній час виконання для циклу тесту показано на вертикальній осі, найменші показники кращі. Інструменти віртуалізації та Linux поширені вздовж горизонтальної осі.

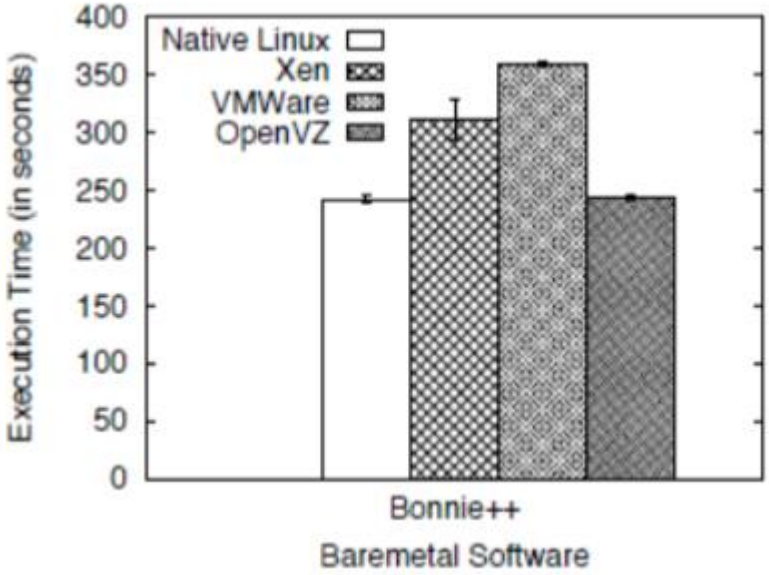


Рисунок 1.7 - Бонні ++ тест

У Linux - а найкращий показник, і OpenVZ досягає подібних результатів результатів, оскільки дисковий доступ в OpenVZ дуже схожий на дисковий доступ у Linux. Xen представив деякі затримки, оскільки він використовує додатковий буфер між dom0 і VM, рівень вводу / виводу, яке додає латентність при читанні з диска. VMware представляє найгірші результати, ймовірно внаслідок додаткових буферів у гіперщитку, які використовуються, щоб передати дані між фізичним диском і віртуальним диском. Тим не менш, дискова робота має незначне значення, оскільки це не впливає на більшу частку операцій по маршрутизатору.

ZFG. Результати для тесту ZFG показано на рисунку 1.8. VMware, Linux і результати OpenVZ були отримані з 10 ступенів. Результати Xen були отримані зі

100 ступенів, щоб мати прийнятне значення похибки. В середньому час виконання на раунди тесту показує вертикальна вісь, найменший показник найкращий. Інструменти віртуалізації та користувацькі Linux розподілені вздовж горизонтальної осі.

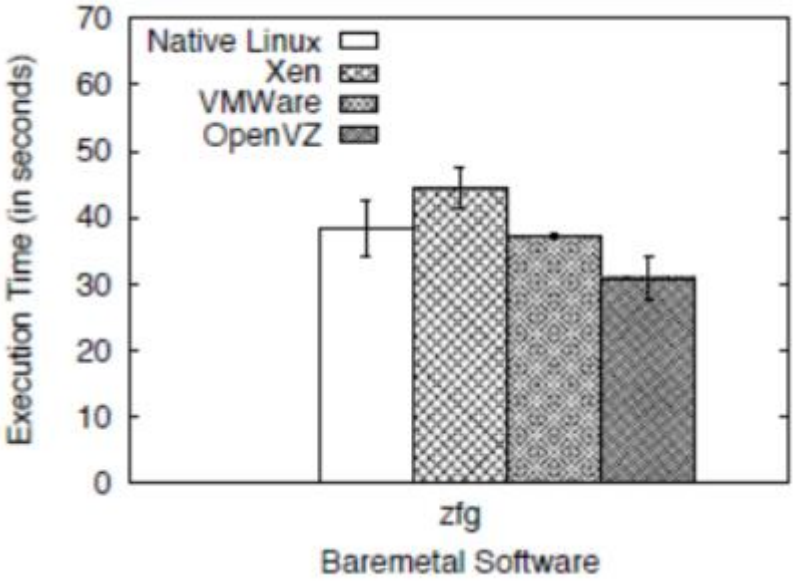


Рисунок 1.8 - Тест ZFG

Як спостерігається на рисунку 1.8, VMware VM проявив трохи кращий результат, ніж Linux, і ми зачитуємо це до буфера, який передає дані між віртуальним диском і реальним диском так, як це робить VM, уявляючи, що дані були вже записані на диск, коли це насправді передається від буфера до фізичного диска. OpenVZ несподівано виграв, показавши найкращий результат. Xen VM показав найгірший результат.

Iperf: потік даних UDP з VM на комп'ютер. Тестування мережі. В цьому тесті визначається ефективність пакета UDP від VM. Результати для еталонного трафіку, що генерує мережі Iperf з VM на комп'ютер тестування для 100 раундів показано на рисунку 1.9. Середня пропускна здатність показана на вертикальній осі, найбільші показники кращі.

Інструменти віртуалізації та Linux поширені вздовж горизонтальної осі.

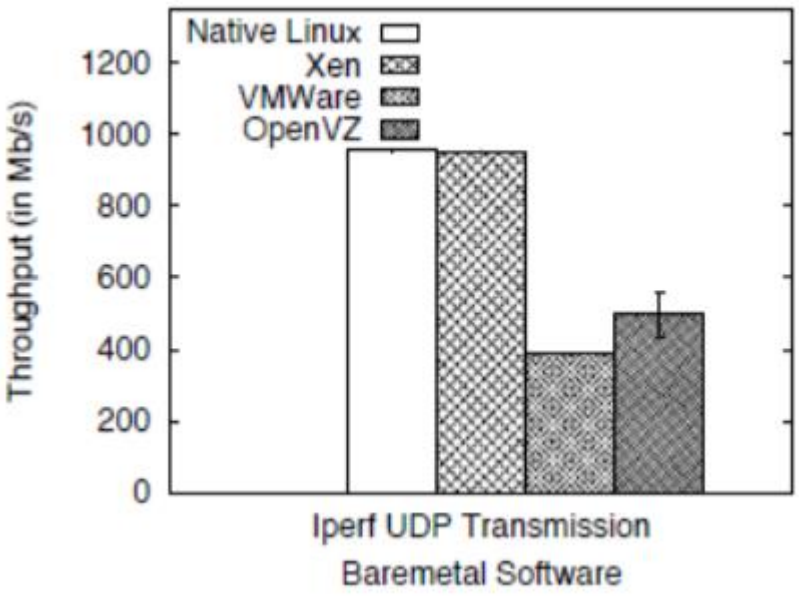


Рисунок 1.9 - Тест Iperf пакета UDP

Як очікувалося Linux на фізичному пристрої досягає найкращих результатів біля теоретичної межі гігабіта адаптера Ethernet. Xen досягає майже такий самий результат роботи, що показує, що механізм введення/виведення здатний постачати дані досить швидко. VMware представляє велику виконавчу деградацію, яка вказує на неефективне впровадження її дефолту мережевим механізмом. Мережевий механізм OpenVZ за дефолтом на основі IP віртуалізації шару виступив погано і не міг використовувати повну пропускну здатність інтерфейсу гігабіта. Це - дуже важливий результат, оскільки пропускна здатність маршрутизатора безпосередньо залежить від його здатності надсилання пакетів, і результати показують, що через віртуалізоване середовище можливо надіслати великі пакети на швидкості майже рівній швидкості фізичного адаптера.

Iperf: потік даних UDP від комп'ютера тестування мережі до VM. В цьому тесті визначається ефективність приймання пакета UDP від VM.

Результати для еталонного трафіку, що генерує мережі Iperf від Комп'ютера Тестування мережі до VM для 100 раундів показуються на рисунку 1.10.

Середня пропускна здатність показана на вертикальній осі, найбільші показники кращі. Інструменти віртуалізації та Linux поширені уздовж горизонтальної осі.

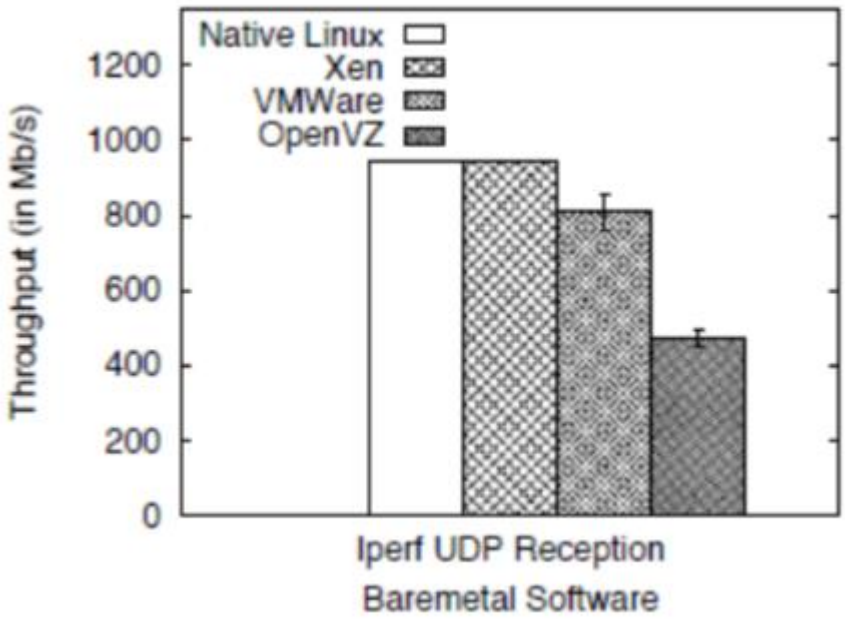


Рисунок 1.10 - Тест Iperf UDP

Результати показують ще раз, що Linux близький до теоретичної максимальної пропускної здатності. Xen ще раз досягає майже фізичних показників. VMware виступив набагато краще в прийомі, ніж у передачі, але все ще гірше, ніж Linux і Xen. Мережевий механізм віртуалізації OpenVZ за дефолтом проявив поганий результат. Результати показують, що Xen може справлятися з прийомом великих пакетів у майже фізичній швидкості.

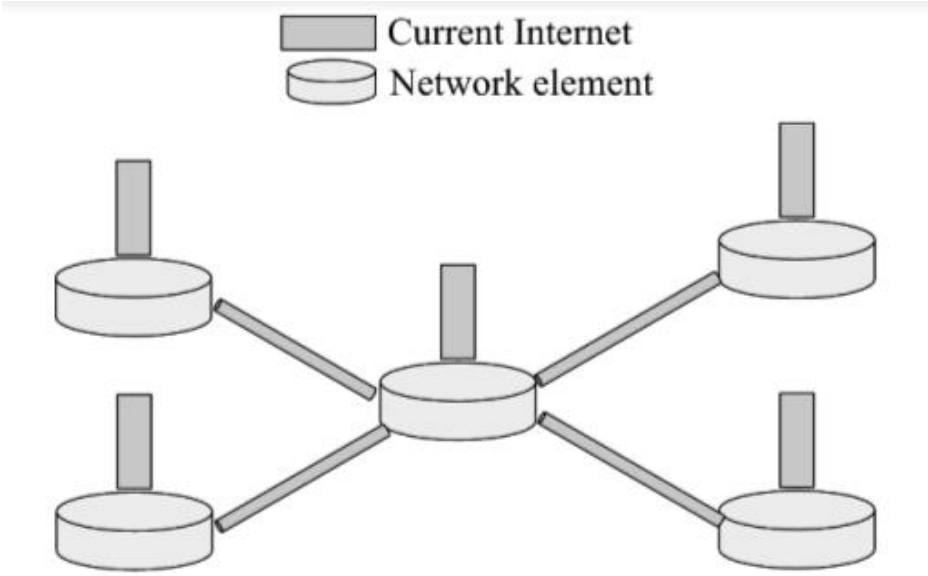
РОЗДІЛ 2. ІНТЕРФЕЙСИ ВІРТУАЛЬНОЇ МЕРЕЖІ

2.1 Віртуальні мережі: продуктивність і тенденції

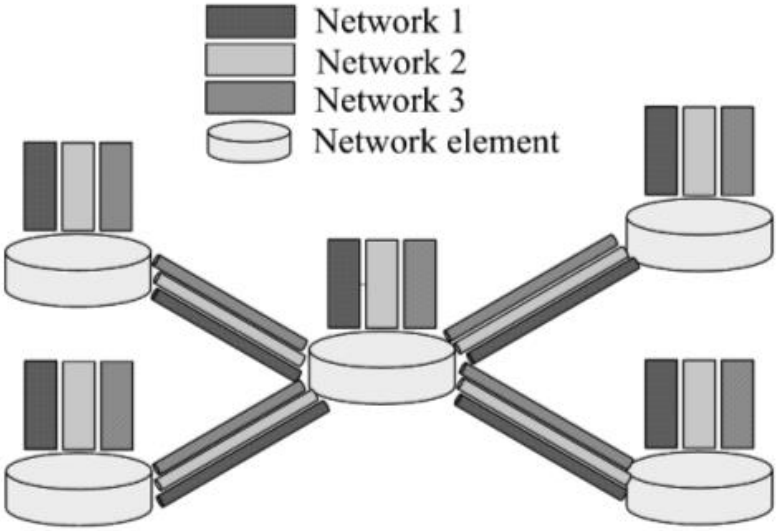
Віртуальні мережеві інтерфейси

Майбутні пропозиції щодо архітектури Інтернету можна розділити на дві основні моделі - моністичну та плюралістичну. У моністичній моделі, показаній на рис. 2.1a), мережа має монолітну архітектуру, яка має бути достатньо гнучкою для підтримки всіх видів додатків. За такого підходу тільки один стек протоколу проходить по фізичній схемі за один раз. З іншого боку, плюралістичний підхід, показаний на рис. 2.1b), заснований на тій ідеї, що Інтернет повинен одночасно підтримувати кілька мереж, що використовують стек протоколів, які відповідають потребам конкретного додатка.

Створення спеціалізованих мереж, призначених для конкретних служб, спрощує розгортання нових додатків, що вимагають виконання таких функцій, як безпека, мобільність або якість обслуговування. В дипломній роботі розкривається підхід, який стверджує, що розробка декількох мереж для забезпечення різних послуг відбувається простіше, ніж проєктування унікальної мережі для обробки всіх послуг одночасно. Таким чином, плюралістичний підхід може бути визначений як «парасолька», заснована на підході «розділяй і володарюй». Якщо дуже складно знайти єдине рішення для покриття всіх можливих вимог, така життєздатна альтернатива може бути створена з кількома різними мережами для підтримки рівнів. Ця характеристика є головним аргументом для плюралістичної архітектури, тому що моністичний підхід не тільки має розв'язати всі відомі Інтернет-проблеми, але також має розвиватися, щоб постійно функціонувати у майбутньому. Крім того, ще однією ключовою перевагою моделі плюралізму є її внутрішня зворотна сумісність, оскільки в поточному режимі в Інтернеті має бути підтримка мереж, що працюють паралельно.



а) Моністична модель



б) Плюралістична модель

Рисунок 2.1 - Мережеві архітектури: моністична модель з одним стек-протоколом і плюралістична модель із кількома стеками протоколів

Плюралістична модель, заснована на ідеї множинних мереж, використовує одну й ту саму інфраструктуру, до того ж ці мережі можуть відрізнятися за форматами пакетів, схемами адресації та протоколами. Оскільки ці мережі використовують один і той самий фізичний мережевий маршрутизатор,

множинний доступ до базового засобу має бути організований монітором. Цей монітор є частиною спеціального програмного забезпечення, який віртуалізує загальний носій у кілька мереж.

Отже, кожна мережа працює так, як якщо б вона використовувала задану кількість фізичних ресурсів. Ці мережі називаються віртуальними мережами і вирішують проблему продуктивності в результаті використання загального фізичного підходу. Це стає додатковим завданням - з'являються накладні витрати плюралістичної архітектури порівняно з моністичною, оскільки додається програмний рівень.

Ґрунтуючись на плюралістичному підході, попередні роботи, такі як дослідження, проведені командою проєкту Horizon, пропонують використовувати дві різні платформи віртуалізації для забезпечення віртуальної мережі: Xen і OpenFlow. Ці платформи розв'язують проблему поділу фізичних ресурсів між кількома віртуальними мережами.

У цьому розділі описується проблема спільного використання мережевого субстрату серед різних віртуальних мереж. У роботі проведено аналіз двох типових підходів до віртуалізації фізичних мереж - Xen і OpenFlow, а також способи використання цих технологій для віртуальних мереж. Основна мета - розкриття плюсів і мінусів мережевої віртуалізації, яка використовується як основа плюралістичної архітектури для майбутнього Інтернету. З цією метою проведено експерименти для оцінки Xen і продуктивності OpenFlow, які виступають у ролі віртуалізованого програмного маршрутизатора. Ґрунтуючись на представлених результатах, у роботі виявлено компроміс між гнучкістю та ефективністю, що вказує на те, що використання даних у загальній площині може бути важливим архітектурним вибором під час розроблення віртуальних мереж. Іншим ключовим моментом є те, що, використовуючи загальні площини даних, Xen і OpenFlow можуть мультиплексувати кілька віртуальних мереж без будь-яких вимірних втрат продуктивності, порівняно зі сценарієм, у якому таку саму швидкість передавання пакетів обробляє один елемент віртуальної мережі.

2.1.1 Підходи до віртуалізації мережі

У дипломній роботі віртуалізацію розглядають як абстракцію ресурсу, що дає змогу розділяти фізичні ресурси на кілька віртуальних ресурсів того самого типу, але з різними можливостями, як це показано на рисунку 2.2. Цю абстракцію часто можна реалізувати програмним шаром, що забезпечує «віртуальні розділені інтерфейси», аналогічні реальним інтерфейсам. Співіснування декількох віртуальних розділів над тим самим ресурсом можливе, тому що рівень віртуалізації відокремлює реальний ресурс і вищеописаний рівень.

На рис. 2.2 показано два приклади віртуалізації: віртуалізація комп'ютерів і віртуалізація мережі. Абстракція віртуалізації комп'ютера реалізується монітором віртуальної машинний (VMM), який надає віртуальним машинам (VM) інтерфейс (тобто рівень абстрагування апаратного забезпечення), вельми схожий на комп'ютерно-апаратний інтерфейс. Цей інтерфейс містить процесор, пам'ять, вхід/вихід (I/O) тощо. Таким чином, кожна віртуальна машина створює враження, що вона працює безпосередньо над фізичним обладнанням, але насправді фізичне обладнання - це кілька віртуальних машин.

Цей поділ ресурсів називається розподілом, тому що віртуальні машини ізольовані: одна віртуальна машина не може втручатися в роботу іншої. Віртуалізація комп'ютерів широко використовується в дата-центрах для запуску декількох серверів на одній фізичній машині.

Крім економії енергії та зниження витрат на технічне обслуговування, найважливішою особливістю цього методу є його гнучкість, що дає змогу кожній віртуальній машині мати свої власні операційні системи, додатки, правила конфігурації та адміністративних процедур. Можливість запуску настроюваного стека протоколів у кожній віртуальній частці є основною мотивацією застосування віртуалізації для мереж. Як показано на рисунку 2.2, віртуалізація мережі є аналогічною до віртуалізації комп'ютерів, у якій спільним ресурсом є мережа. Ця концепція не є новою, і її було використано у віртуальних приватних мережах (VPN) та у віртуальних локальних мережах (VLAN). Нині, однак,

існують нові методи, що дають змогу навіть віртуалізувати маршрутизатор. У цьому разі кожна віртуальна частина маршрутизатора може реалізувати стек мережевих протоколів, що налаштовується.

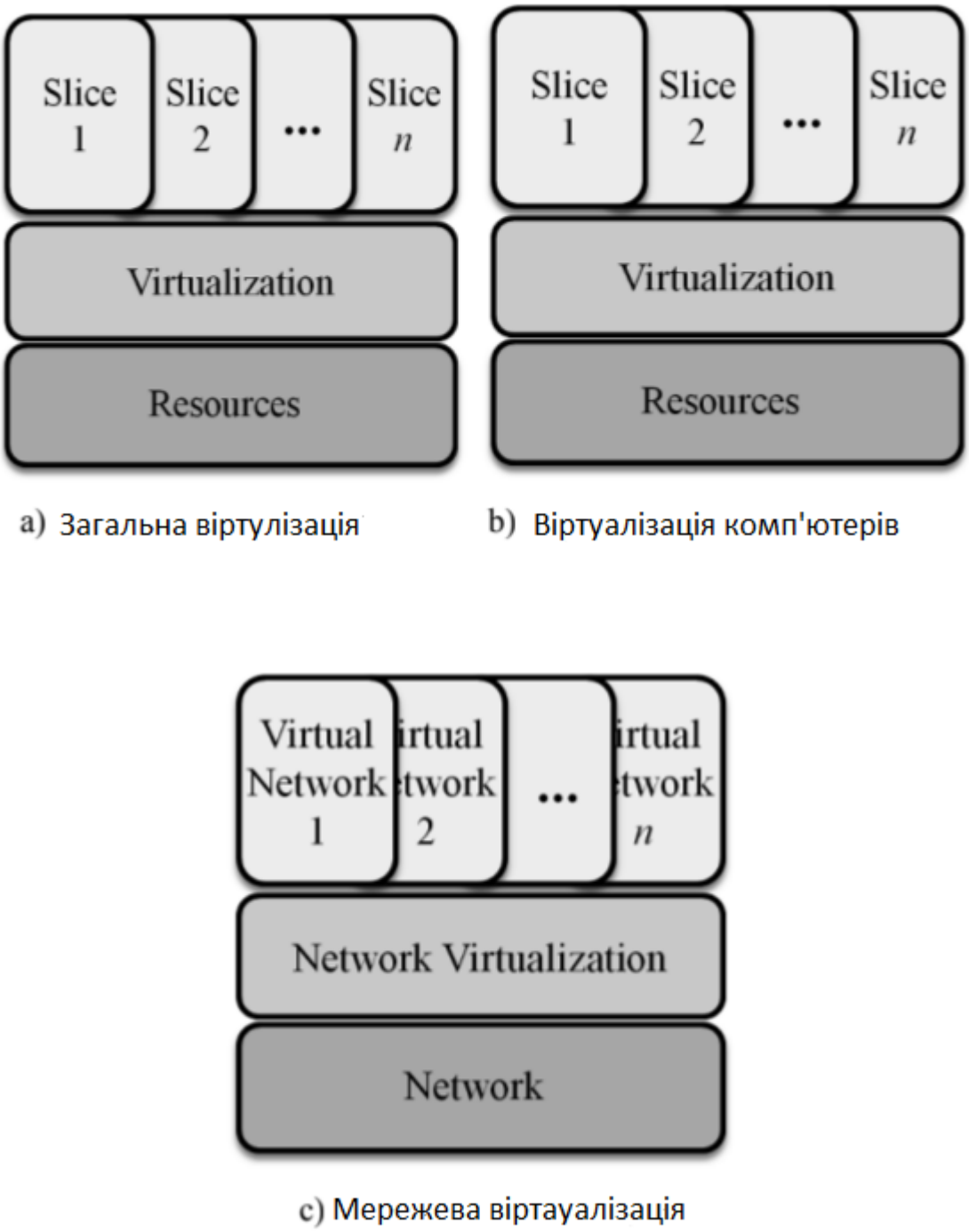
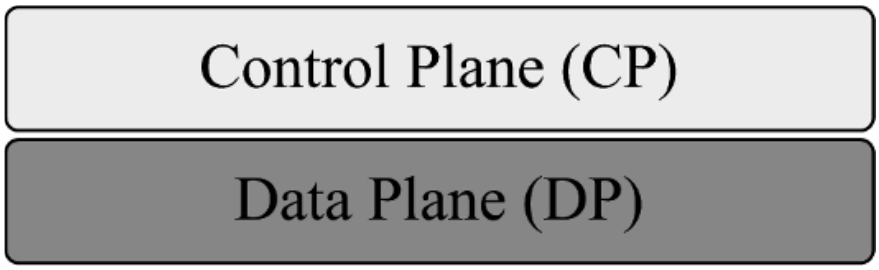


Рисунок 2.2 - Віртуалізація: «розділені» ресурси, віртуальні машини і віртуальні маршрутизатори

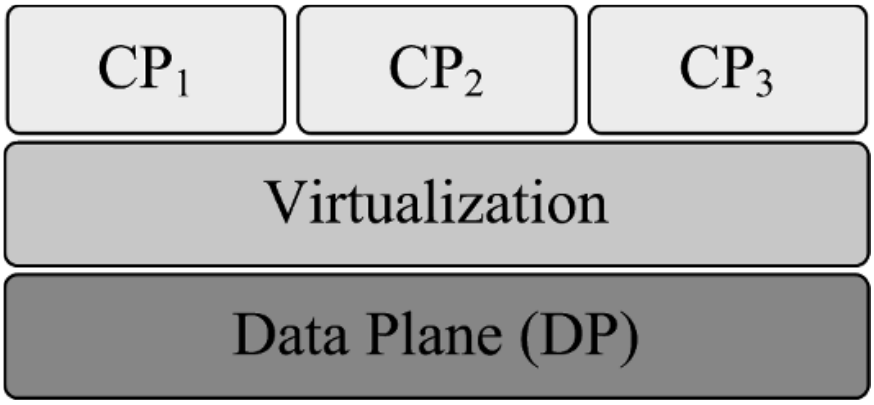
Підходи до зіставлення віртуалізації мережі на рівні абстракції показують, що можна проілюструвати положення шару віртуалізації в архітектурній моделі. На рисунку 2.3 порівнюються два основні підходи для віртуалізації елемента мережі. На рис. 2.3 a) показано звичайну архітектуру мережевого елемента з єдиною панеллю керування та даними. У маршрутизаторі площина управління відповідає за запуск програмного забезпечення мережевого управління, таких як протоколи маршрутизації (наприклад, Routing Information Protocol (RIP), Open Shortest Path First (OSPF) і Border Gateway Protocol (BGP)) і протоколи управління мережею (наприклад, Internet Control Message Protocol (ICMP)), тоді як площина даних відповідає за реалізацію таблиць пересилання та апаратні канали передавання даних. Віртуалізація процедури маршрутизації означає вставку віртуалізації на деякий рівень архітектури мережевих елементів, що дає змогу співіснування декількох віртуальних мережевих елементів над одним фізичним мережевим елементом. Передбачається наявність рівня віртуалізації між контролем і площиною даних, де система управління є віртуальною, як це показано на рисунку 2.3 b). У цьому випадку площина даних використовується всіма віртуальними мережами, і кожна віртуальна мережа запускає власне програмне забезпечення управління. Порівняно з традиційною архітектурою мережі, цей підхід значно покращує мережеву програмованість, оскільки він дає змогу запускати кілька протоколів, які можна налаштовувати, замість одного стека протоколу за замовчуванням. Наприклад, можна програмувати різні стеки протоколів для мережі 1, мережі 2 і мережі 3, як показано на рис. 2.3 b). У другому підході мережевої віртуалізації панелі управління і дані віртуалізовані, як показано на рисунку 2.3 c). У цьому разі кожен елемент віртуальної мережі реалізує свою власну площину даних поруч із площиною керування, покращуючи тим самим програмованість мережі. Цей підхід дає змогу налаштовувати площини даних завдяки пересиланню пакетів, оскільки площина даних більше не призначена для загальних завдань.

Варто зазначити, що тільки віртуалізацією панелі управління можна визначити більшу кількість підходів залежно від рівня ізоляції площини даних,

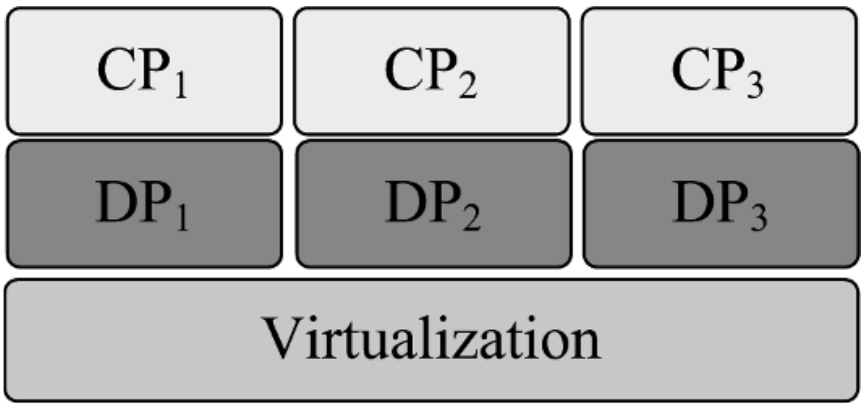
які розділяються між віртуальною мережею та її елементами. Якщо потрібна сильна ізоляція, то кожна віртуальна площина повинна мати доступ тільки до її розділу площини даних і не може втручатися в інші. З іншого боку, якщо вся площина даних розподіляється між віртуальними керуючими площинами, тоді можливо, що віртуальна панель управління буде інтерферувати з іншими віртуальними площинами управління. Наприклад, одна віртуальна контрольна площина може заповнювати всю таблицю пересилання своїми власними записами, що може призвести до того, що пакети інших віртуальних машин можуть бути загублені.



a) Звичайна модель



b) Плюралістична модель тільки з віртуалізованою моделлю віртуальної машини



c) Плюралістична модель із віртуалізованою панеллю управління і дата планами

Рисунок 2.3. Мережеві моделі

2.1.2 Технології мережевої віртуалізації

У цьому розділі детально розглянуто дві технології, які можуть бути використані для віртуалізації мережі: Xen і OpenFlow. Xen - це VMM (монітор віртуальних машин) з відкритим вихідним кодом, також званий гіпервізором, який працює на основній апаратній платформі. Архітектура Xen складається з одного VMM, розташованого над фізичним обладнанням і декількома доменами, одночасно працюючи над гіпервізором, званим віртуальною машиною, як показано на рисунку 2.4.

Кожна віртуальна машина може мати свою власну операційну систему і додатки. VMM контролює доступ декількох доменів до обладнання, а також керує розподілом ресурсів між цими доменами. Отже, одним з основних завдань VMM є ізолювання різних віртуальних машин, тобто робота однієї VM не повинна впливати на продуктивність інших. Крім того, всі драйвери пристроїв зберігаються в ізольованому домені драйверів, який називається Domain 0 (dom0) для забезпечення надійної та ефективної апаратної підтримки. Домен 0 має особливі привілеї порівняно з іншими доменами, званими непривілейованими доменами (domUs), оскільки він має повний доступ до фізичного машинного обладнання. Непривілейовані домени, також звані користувацькими доменами, мають віртуальні драйвери і працюють так, як якщо б вони могли безпосередньо

звертатися до обладнання. Проте, ці віртуальні драйвери взаємодіють з dom0, щоб мати доступ до фізичного обладнання.

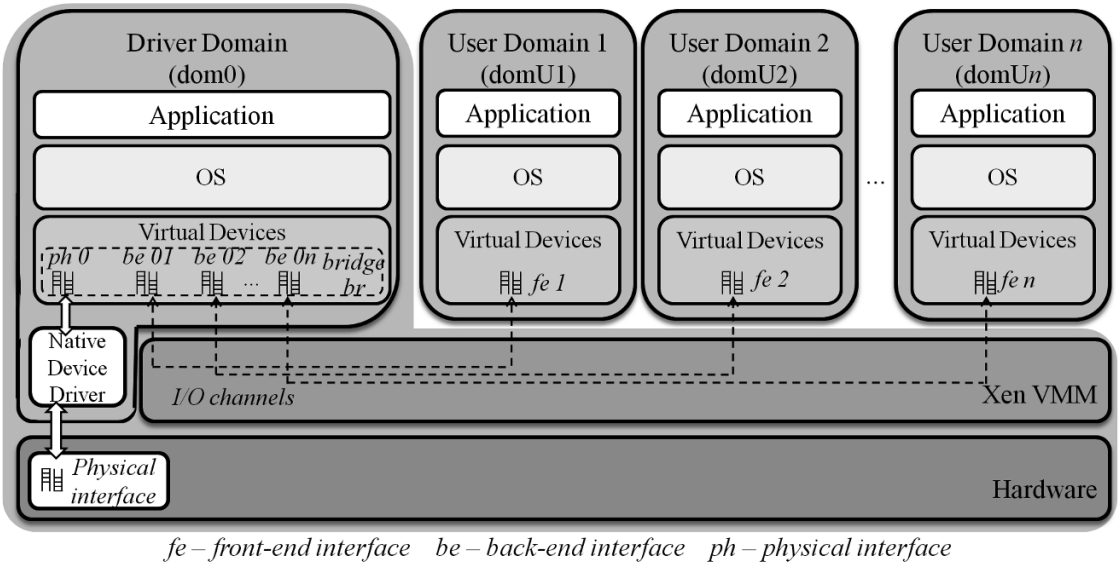


Рисунок 2.4 - Архітектура Xen: монітор віртуальної машини (VMM), домен драйвера (dom0) і доменів користувачів (domU)

Xen віртуалізує один фізичний мережевий інтерфейс за допомогою демультиплексування вхідних пакетів від фізичного інтерфейсу до domUs і, навпаки, мультиплексування вихідних пакетів, що генеруються цими domU. Ця процедура - так звана мережева віртуалізація введення-виведення - працює так. Домен 0 безпосередньо отримує доступ до пристроїв введення-виведення, використовуючи свої власні драйвери пристроїв, а також виконує операції введення-виведення від імені domUs. З іншого боку, domUs використовують віртуальні пристрої введення/виведення, керовані віртуальними драйверами, для запиту dom0 доступу до пристроїв, як показано на рисунку 2.4. Кожен domU має свій власний віртуальний мережевий інтерфейс, званий зовнішнім інтерфейсом, необхідним для комунікаційної мережі. Внутрішні інтерфейси створюються в dom0, кореспондують із кожним зовнішнім інтерфейсом у domU і діють як проксі для віртуальних інтерфейсів у dom0. Внутрішні та зовнішні інтерфейси під'єднані один до одного через канал вводу/виводу, який використовує механізм нульової копії для перепризначення фізичної сторінки, що містить пакет у цільовому

домені. Таким чином, пакети обмінюються між back-end і front-end інтерфейсами. Зовнішні інтерфейси сприймаються системою, що працюють на domU як реальні інтерфейси. Проте, інтерфейси в dom0 підключені до фізичного інтерфейсу, а також до кожного через віртуальний мережевий міст.

Це - стандартна архітектура, яка використовується Xen, і вона називається «мостовим режимом». Таким чином, як канал введення/виведення, так і мережевий міст встановлюють канал зв'язку між віртуальними інтерфейсами, створеними в domUs, і фізичним інтерфейсом. Різні елементи віртуальної мережі можна реалізувати з використанням Xen, оскільки ця технологія дає змогу кільком віртуальним машинам працювати одночасно на одному й тому самому обладнанні, як показано на рисунку 2.5. У цьому випадку кожна віртуальна машина запускає віртуальний маршрутизатор, який має власні панелі управління і даних, оскільки рівень віртуалізації Xen низькорівневий.

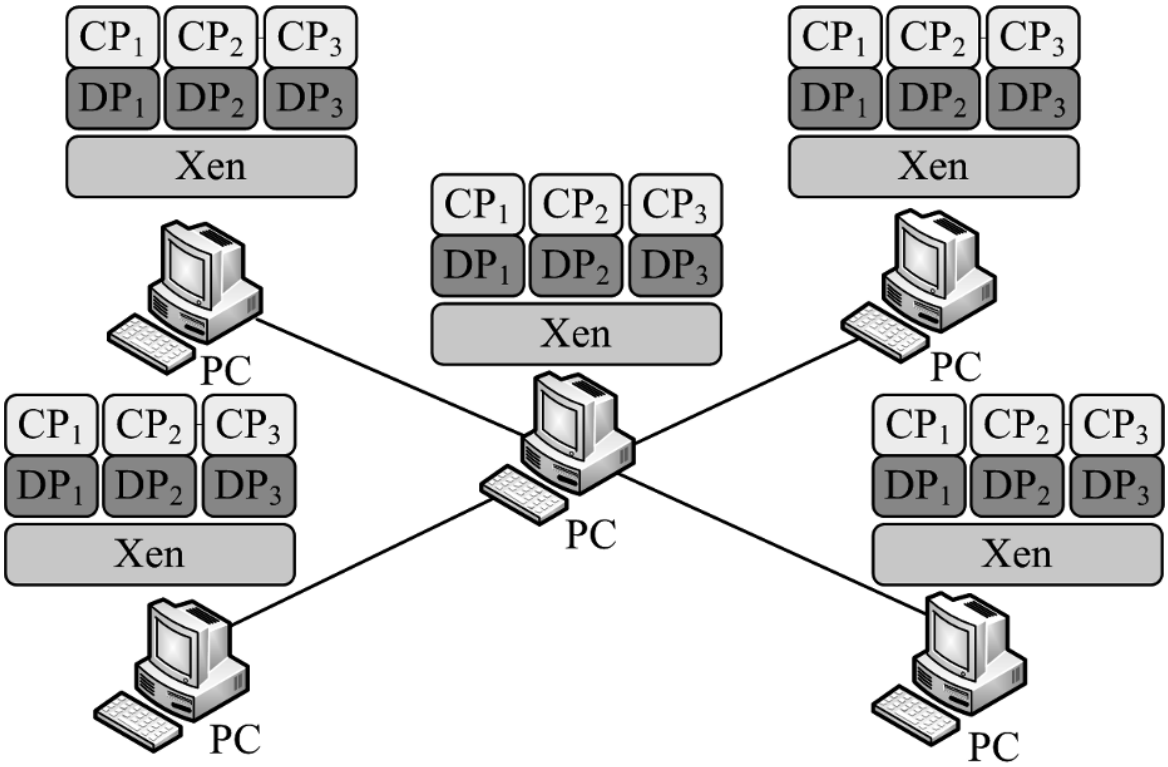


Рисунок 2.5 - Віртуальна мережа Xen: одна панель даних і управління на віртуальний маршрутизатор

OpenFlow дає змогу використання комутаційних приміщень у кампусі університету не лише для виробничої мережі, а й для експериментальних мереж.

Проєкт OpenFlow, запропонований Стенфордським університетом, спрямований на створення віртуальних середовищ для інновацій паралельно з виробництвом мережі з використанням мережевих елементів, таких як комутатори, маршрутизатори, точки доступу та персональні комп'ютери. OpenFlow представляє нову архітектуру середовищ для віртуальної мережі. Ключовою ідеєю є фізичний поділ управління та дата планів. Тому різні мережеві елементи виконують пересилання пакетів (панель даних) і процедуру управління мережею (панель управління).

Віртуалізація елементів пересилання здійснюється за допомогою спільної таблиці потоків, яка являє собою панель даних, тоді як усі площини управління централізовані у вузлі, який називається контролером, запуску додатків, які керують кожною віртуальною мережею. Приклад роботи OpenFlow показано на рисунку 2.6.

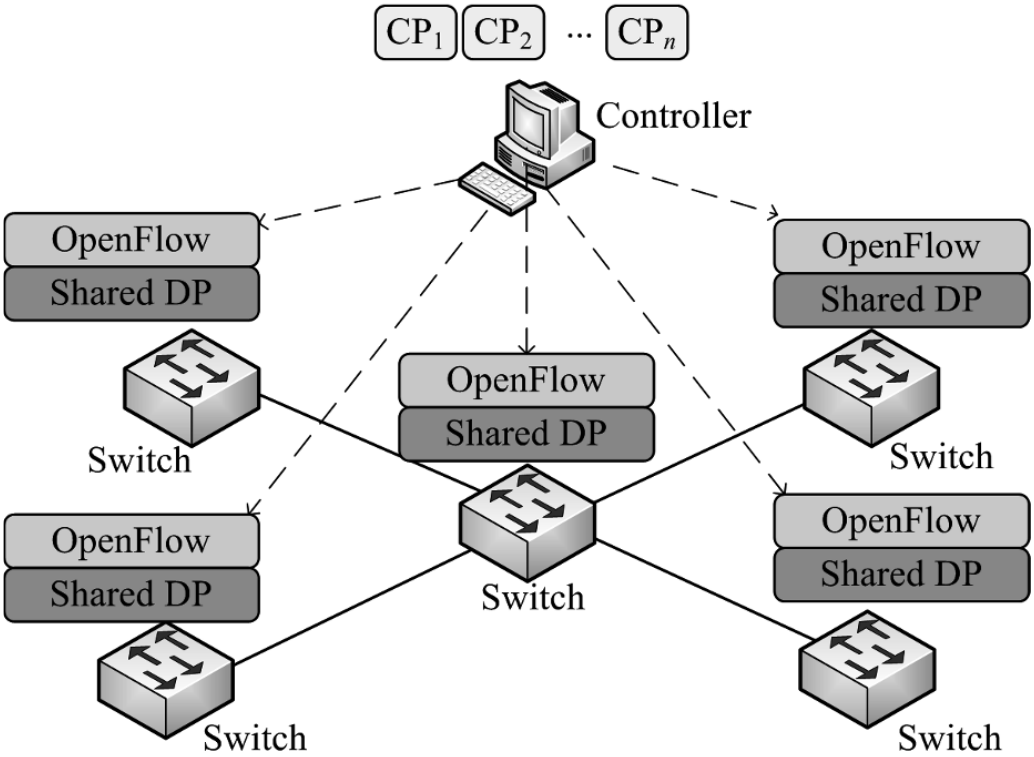


Рисунок 2.6 - Віртуальна мережа OpenFlow: загальна панель даних для кожного вузла і всі площини управління в одному вузлі

Протокол OpenFlow визначає зв'язок між пересиланням вузлів і мережевого контролера. Він заснований на створенні безпечного каналу між кожним вузлом

пересилання і контролером, який використовує цей канал для моніторингу та налаштування вузлів пересилання. Щоразу, коли новий пакет досягає елемента переадресації, перші біти пакета пересилаються в контролер, який встановлює шлях для цього пакета в обраних елементах пересилання. Контролер також може встановлювати дії нормального опрацювання для потоку, що підлягає пересиланню відповідно до звичайної маршрутизації рівня 2 (L2) і рівня 3 (L3), нібито OpenFlow не існує. Саме з цієї причини OpenFlow можна використовувати паралельно з виробничою мережею, не зачіпаючи поточний трафік. Панель даних в OpenFlow - це таблиця потоків, описувана полями заголовка, лічильників і дій. Площина заголовка являє собою структуру з 12 частин, яка описує заголовок пакета, як показано на рисунку 2.7. Ці площини визначають потік, задаючи значення для кожної площини або використовуючи шаблон для встановлення тільки підмножини площин. Таблиця потоків також підтримує використання масок підмережі, якщо обладнання також підтримує це. Ця структура з 12 частин дає високу гнучкість для пересилання пакетів, оскільки потік може бути переспрямований на основі не тільки за адресою IP-адреси одержувача, як у звичайному протоколі управління передачею (TCP)/IP, але також за портом TCP, MAC-адресою і т. д. Оскільки потоки можуть бути встановлені на основі адрес рівня 2, елементи пересилання OpenFlow також називаються комутаторами OpenFlow. Це, однак, не обов'язково має на увазі пакет пересилання в шарі-2. Однією з майбутніх цілей OpenFlow є обробка призначених для користувача полів заголовка, що означає, що заголовок пакета не буде описаний фіксованими полями, але комбінацією полів, заданих адміністраторами віртуальних мереж. Це дасть OpenFlow можливість пересилання пакетів, що належать мережам, які використовують будь-який стек протоколу.

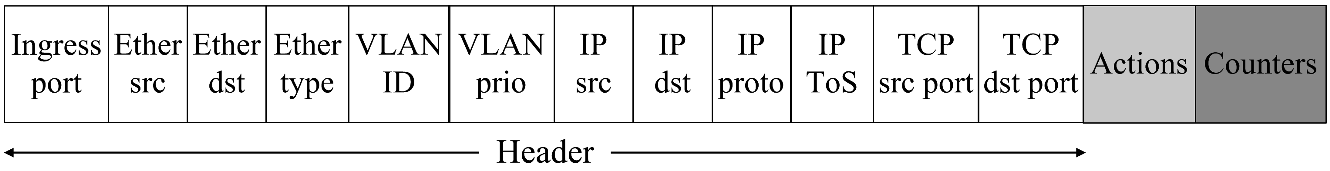


Рисунок 2.7 - Запис потоку в мережі OpenFlow

Після полів заголовка в описі потоку йдуть лічильники, які використовуються для контролю вузлів. Лічильники обчислюють дані, такі як тривалість потоку і кількість пересилаються байтів. Останні поля в описі потоку - дії, які являють собою набір інструкцій, що можуть виконуватися за кожним пакетом певного потоку в елементах, що пересилаються. Ці дії включають не тільки перемикання вхідного пакета на порт призначення, а й зміну заголовків, таких як ідентифікатор VLAN і адреси, і джерела, і одержувача. Контрольний вузол є центральним елементом мережі і може взаємодіяти з усіма вузлами для налаштування своїх таблиць потоків. Контролер запускає мережеву операційну систему, яка забезпечує управління віртуальною мережею за допомогою основних функцій для налаштування мережі. Отже, контролер в OpenFlow працює як інтерфейс між мережею додатків і передачею елементів, забезпечуючи основні функції доступу до пакетів з потоку і для моніторингу вузлів. OpenFlow працює з будь-яким контролером, сумісним із протоколом OpenFlow, наприклад, такий як мережа операційної системи (NOX). У цьому випадку кожна площина управління складається з набору додатків, що працюють над NOX. Отже, віртуальна мережа в OpenFlow визначається своєю площиною управління, яка являє собою набір додатків, що працюють над контролером, і відповідних їм потоків, як показано на рисунку 2.8.

Використовуючи модель одного контролера, можна створити безліч віртуальних мереж. Звернемо увагу на те, що різні додатки, які працюють на одній і тій самій ОС, не ізольовані. У результаті, якщо в одному додатку є помилка, вона може зупинити контролер, пошкодивши всі інші віртуальні мережі. FlowVisor - це інструмент, який використовується з OpenFlow, щоб дозволити різним контролерам працювати на тій самій фізичній мережі. FlowVisor працює як проксі-сервер між переданими елементами і контролерами, припускаючи, наприклад, один контролер на мережу. Використовуючи цю модель, можна гарантувати, що збої в одній віртуальній мережі не впливають на інші.

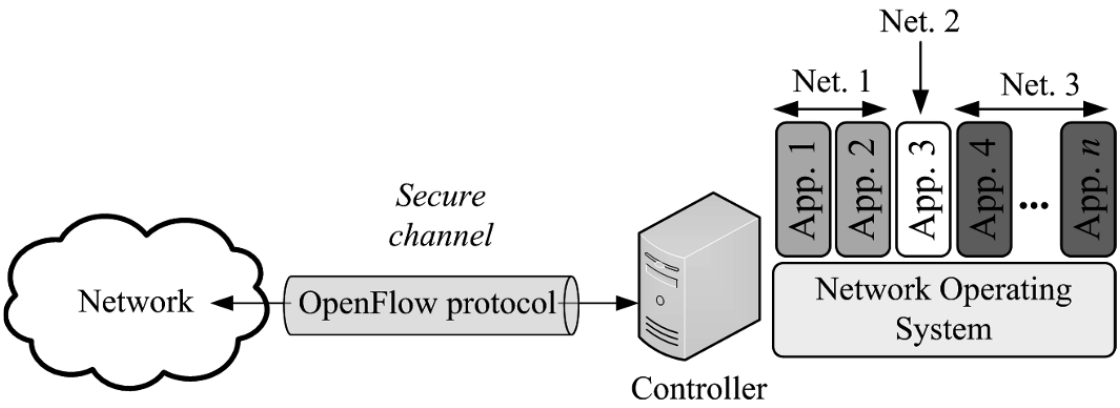


Рисунок 2.8 - Централізована модель контролера OpenFlow

OpenFlow надає гнучку інфраструктуру, засновану на ідеї розподілених елементів пересилання, що пропонують базові функції для мережі та централізованих панелей управління. Використовуючи цю інфраструктуру, можна розрізати фізичну мережу на кілька віртуальних мереж. В OpenFlow, створення мережі - це просто створення деякого набору додатків у контролері. Нові мережеві потоки будуть створені відповідно до пакетів, що входять у мережу. OpenFlow також надає гнучку інфраструктуру для перерозподілу мережевих ресурсів. Перерозподіл мережі в OpenFlow означає тільки перепрограмування таблиці потоків кожного вузла, що бере участь. Це проста операція для контролера, тому що він знає, де розташовані фізичні пристрої і як вони пов'язані.

2.2. Xen прототип

Прототип Xen розроблено в лабораторії Grupo de Teleinformática e Automação (GTA) для експериментів із новими інтерфейсами. У цьому розділі роботи описуються інтерфейси, пов'язані з Xen прототипами, розробленими відповідно до архітектури, описаної на рисунку 2.9. Панель макетування запитує служби на сервері віртуальної машини (VMS). Ці послуги можуть бути пов'язані з розпізнаванням або дією на інфраструктуру. VMS виконує необхідну дію і відправляє відповідь на площині пілотування. Для спрощення реалізації площини

макетування інтерфейсів, пропонованих VMS, мають бути чітко визначені, а платформа незалежна.

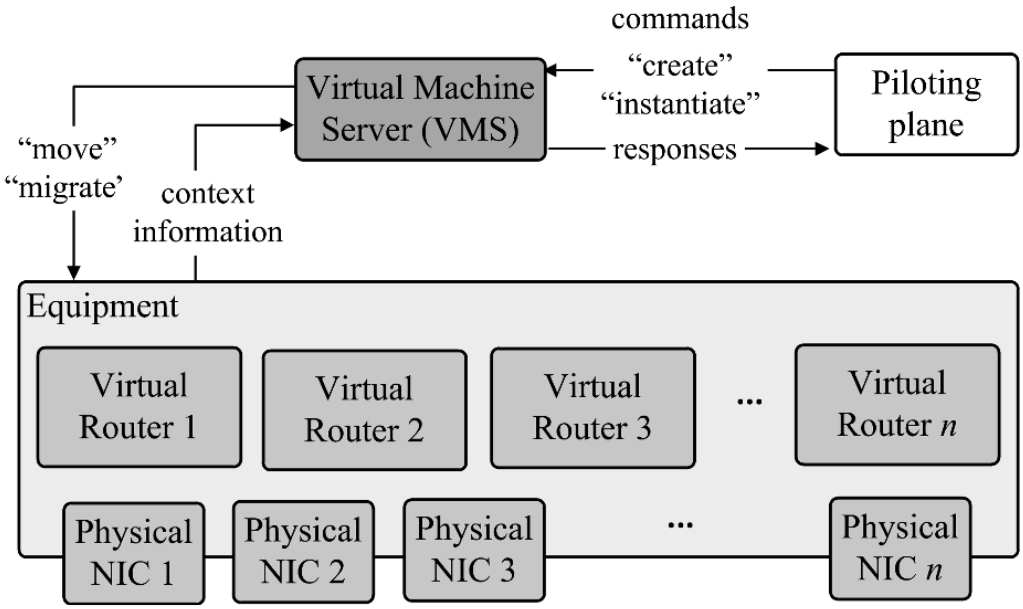


Рисунок 2.9 - Архітектура прототипу Horizon Xen

VMS надає набір сервісів для управління віртуальними мережами та віртуальними маршрутизаторами. Ця система забезпечує віртуальні маршрутизатори за запитом для відповідності певним вимогам. Отримавши запити на нові віртуальні мережі, сервер створює правильну кількість віртуальних машин і розгортає їх у певних вузлах фізичної мережі. Крім того, цей сервер може брати участь у діяльності мережевої адміністрації.

Сервер може бути реалізований з використанням веб-служб і може відповідати на запити, використовуючи протокол простого доступу до об'єктів (SOAP). Веб-сервер може бути представлений Apache Tomcat. Цей підхід спрощує створення гетерогенних клієнтів для VMS і зменшує складність додавання нових функцій. Кожна послуга виробляється загальнодоступним методом, що моделюється як клас VirtualMachineServer. Панель макетування може діяти в мережі та самостійно виконувати деякі зміни, як це пропонується в майбутніх інтернет-проектах. Наприклад, ми можемо зменшити навантаження перевантаженої фізичної машини методом перенесення однієї з машин на інший фізичний вузол. У цьому випадку панель макетування відправляє команду в VMS

з використанням протоколу SOAP для запиту міграції віртуальної машини. Потім VMS використовує Libvirt - віртуалізовані системні бібліотеки управління для виконання операції міграції. Серед доступних послуг у прототипі Xen є ті, за допомогою яких можна керувати кількома віртуальними мережами. У наступних, типових послугах, що надаються в тестовому стенді на основі Xen, узагальнено методи для плюралістичної архітектури разом з описом їхніх основних цілей.

CreateVirtualMachine. Цю службу потрібно викликати щоразу, коли нові віртуальні машини потрібно створити на вузлі мережі, щоб допомогти для забезпечення роботи мережі прототипу Xen за допомогою операцій людини. Цей інтерфейс може замінити площину макетування і може мати доступ до всіх мережевих завдань адміністрування.

CreateVirtualNetwork. Ця служба створює набір віртуальних машин на деяких фізичних вузлах мережі. Крім того, сервер VM повинен відображати створений віртуальний мережевий інтерфейс на зазначену фізичну мережу.

DestroyVirtualMachine. Ця служба знищує віртуальну машину. Знищена віртуальна машина не може бути повторно використана в майбутньому.

RegisterNodes. Ця служба може використовуватися для реєстрації наявних вузлів у мережі, тобто імені, відкритого ключа та IP-адреси віртуальних машин, які будуть збережені на сервері VM.

GetPhysicalServerStatus. Ця служба отримує список основних інформацій про фізичний сервер. Поточний список містить кількість ЦП, кількість ядер, обсяг оперативної пам'яті, обсяг вільної пам'яті, ім'я хоста, номери та імена активних віртуальних доменів.

GetRegisteredNodes. Ця служба повертає список зареєстрованих вузлів на сервері VM.

GetVirtualMachineStatus. Ця служба повертає список основної інформації про віртуальну машину. Даний список містить ім'я віртуальної машини, поточний об'єм оперативної пам'яті, загальну RAM-пам'ять, яку можна використати, поточну кількість віртуальних процесорів (VCPU), максимальну кількість VCPU,

що VM може використовувати, час процесора і поточний стан віртуальної машини.

MigrateVirtualMachine. Ця служба переносить віртуальну машину з одного фізичного хоста на інший у тій самій мережі.

SanityTest. Ця послуга є перевіркою працездатності для сервера VM. Клієнт відправляє рядок на сервер, і сервер відправляє назад той самий рядок.

ShutdownVirtualMachine. Ця служба вимикає віртуальну машину. У цьому випадку, VM можна використовувати в майбутньому.

TopologyDiscover. Ця служба створює матрицю із сусідніми вузлами на фізичних і віртуальних мережах. Існує одне обмеження на цю послугу: вузол повинен бути зареєстрований на сервері, використовуючи службовий регістр Nodes, щоб бути частиною фізичної топології, а також мати віртуальні машини у віртуальних топологіях.

GetVirtualMachineSchedulerParameters. Ця служба запитує гіпервізор для планувальника CPU, і параметри віртуальної машини.

Клієнт сервера віртуальних машин. Для спрощення розвитку клієнта клас може бути запропонований з урахуванням кожної нової послуги, що додається на сервері. Для кожної нової послуги використовується метод створення корисного навантаження повідомлень, що додається в клас клієнта. У наступному, можливі повідомлення перераховані з їх відповідними корисними навантаженнями, реалізованими в класі клієнта, для взаємодії з VMS. Ці повідомлення пропонуються в проекті Horizon.

CreateVirtualMachinePayload. Цей метод створює корисне навантаження для запиту про створення віртуальної машини на основі імені фізичної машини, в якому вона розміщена.

IP-адреса і необхідний обсяг оперативної пам'яті. Метод повертає Extensible Markup Language (XML), представлений об'єктом класу OMElement, з результатом операції, який може бути успішним або невдалим.

CreateVirtualNetworkPayload. Цей метод створює корисне навантаження для запиту про створення віртуальної мережі на основі списку фізичних імен машин,

на яких розміщуватимуться нові віртуальні машини, списку з іменами нових віртуальних машин, списку з бажаними IP-адресами, а також списку з потрібною RAM-пам'яттю та фізичними мережевими інтерфейсами, які будуть відображені на нових віртуальних мережевих інтерфейсах, створених на віртуальних машинах. Метод, що повертає XML-повідомлення, представлений об'єктом класу OMElement, а результатом операції може бути успішним або невдалим.

DestroyVirtualMachinePayload. Цей метод створює корисне навантаження для запиту про знищення віртуальної машини на основі імені фізичної машини, яка містить віртуальну машину та ім'я віртуальної машини. Метод повертає XML повідомлення, представлене об'єктом класу OMElement, а результат може бути успішним або невдалим.

GetPhysicalServerStatusPayload. Цей метод створює корисне навантаження для отримання статусу фізичного сервера, заснованого на імені фізичної машини. Метод повертає XML-повідомлення, представлене об'єктом класу OMElement, з результатом, який може бути успішним або невдалим, кількістю процесорів, кількістю ядер, загальним об'ємом оперативної пам'яті, кількістю вільної оперативної пам'яті, ім'ям хоста, а також номером та ім'ям активних віртуальних доменів.

GetVirtualMachineStatusPayload. Цей метод створює корисне навантаження для отримання статусу віртуальної машини, що ґрунтується на імені фізичного комп'ютера, на якому розміщено віртуальну машину, та імені віртуальної машини. Метод повертає XML-повідомлення, представлене об'єктом класу OMElement, з результатом операції, який може бути успішним або невдалим, ім'ям віртуальної машини, поточним об'ємом оперативної пам'яті, загальною RAM-пам'яттю, яку можна використовувати, поточною кількістю VCPU, максимальною кількістю VCPU, які VM може використовувати, часом процесора, що використовується, і поточним станом віртуальної машини.

MigrateVirtualMachinePayload. Цей метод створює корисне навантаження для перенесення віртуальної машини на основі імені вихідної фізичної машини, імені кінцевого фізичного комп'ютера, імені віртуальної машини та рядка з

вказівкою, чи є операція в реальному часі, тобто, у разі, якщо станеться міграція без переривання програм, запущеної на віртуальній машині. Метод повертає XML-повідомлення, представлене об'єктом класу OMElement, з результатом операції, який може бути або успішним, або невдалим.

TopologyDiscoverPayload. Цей метод створює корисне навантаження для виявлення топології. Сервіс не має параметрів. Метод повертає XML-повідомлення, представлене об'єктом класу OMElement, із результатом фізичної та віртуальної топологій.

Графічний інтерфейс користувача (GUI). GUI прототипу Xen може отримати доступ до VMS через запити командного рядка. Усі служби, які використовують користувачі GUI, отримують доступ через комунікаційний інтерфейс Веб-сервісу. Клієнт командного рядка отримує необхідні служби і параметри від GUI.

2.3 OpenFlow прототип

Прототип OpenFlow було розгорнуто в лабораторії GTA для експериментів із новими інтерфейсами. У цьому розділі роботи описуються компоненти, пов'язані з прототипом OpenFlow. Дотримуючись тієї ж ідеї прототипу Xen, прототип OpenFlow також може ґрунтуватися на веб-сервісах. Зв'язок між ядром прототипу і зовнішніми додатками використовує HTTP (HyperText Transfer Ptotocol) для обміну XML-повідомленнями. Щоб виміряти мережу OpenFlow можна використовувати датчики. Ці датчики здебільшого є лічильниками, встановленими на комутаторах, доступних через протокол OpenFlow, або у вигляді інформації на таблиці OpenFlow, такої як кількість записів потоку та інші статистичні дані.

Додатки NOX збирають інформацію про датчики і роблять їх доступними як Веб-сервіс. На рис. 2.10 показано архітектуру прототипу OpenFlow. NOX контролер є базою для додатків OpenFlow. Контролер NOX забезпечує протокол OpenFlow і реалізацію безпечного каналу для додатку.

Список додатків, що працюють на контролері NOX, разом з їхніми основними цілями описані нижче.

Додаток статистики: цей додаток збирає статистику про комутатори та перетворює їх на XML-повідомлення.

Додаток виявлення: цей додаток виявляє мережеву топологію і описує це як XML-повідомлення.

Додаток SpanningTree: цей додаток реалізує остовне дерево - алгоритм, який дає змогу уникнути виникнення мережевих циклів.

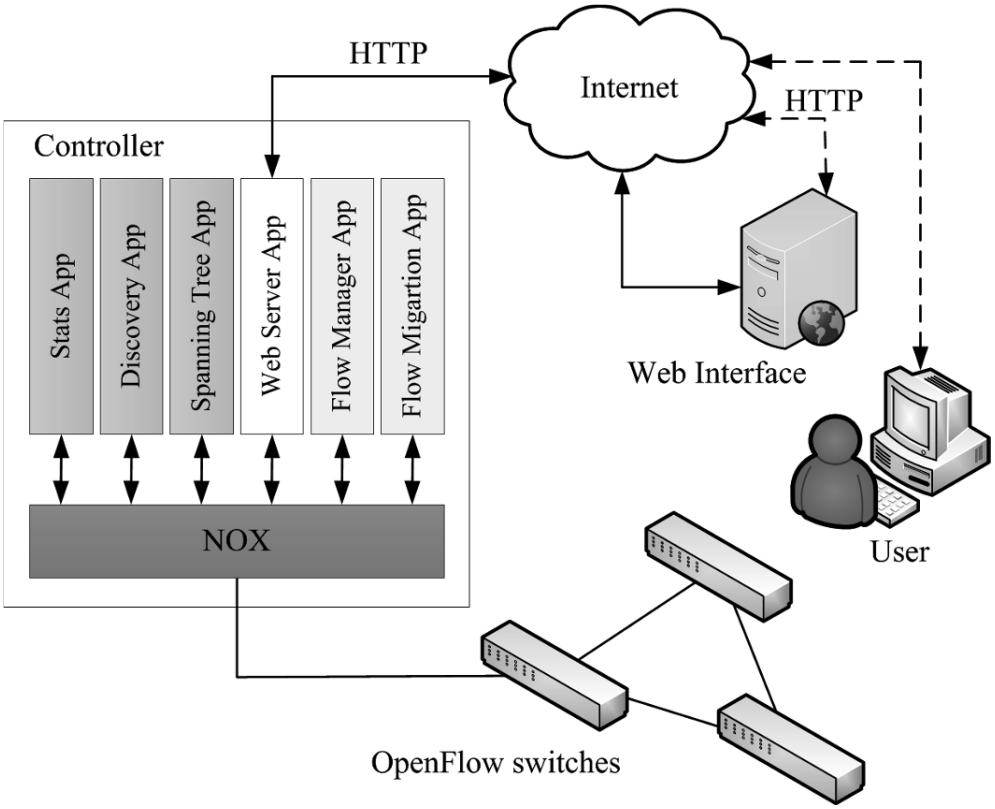


Рисунок 2.10 - Додатки OpenFlow, NOX і взаємодії між агентами

Основними критеріями під час вибору набору технологій були: наявність досвіду роботи з обраною мовою програмування, а також зручність фреймворків для розроблення веб-додатків. Як мову програмування для серверної частини програми було обрано мову програмування Python. Як фреймворк було обрано Django через його популярність і бажання розробника краще вивчити цю технологію.

Додаток FlowManager: цей додаток реалізує зміни потоку як додавання, модифікація та видалення потоків.

Додаток FlowMigration: цей додаток реалізує зміни потоку, перенесення потоку з одного шляху на інший.

Додаток WebServer: цей додаток забезпечує інтеграцію між функцією додатку та контролером NOX. Додаток WebServer реалізує протокол HTTP для забезпечення інтерфейсу між додатками контролера NOX і зовнішніми додатками. Отже, додаток контролера NOX обробляє HTTP-запити, перетворює їх на виклик методу додатка і виконує метод. Прототипом, що використовується для експериментів, є клієнт для додатка WebServer. Клієнт - це ще один веб-сервер, який дозволяє адміністраторам керувати мережею OpenFlow.

Веб-сервер OpenFlow. Додаток WebServer - це додаток NOX, який відповідає за надання веб-інтерфейсу для інших додатків NOX. Додаток WebServer реалізує концепцію веб-служби, в якій функціональність інших додатків може бути доступною за HTTP-запитом, і повертає XML-повідомлення. Реалізація цього додатка заснована на мережі NOX, серверний додаток за замовчуванням можна налаштувати для роботи на порту 8080, можливе прослуховування HTTP-запитів. Усі HTTP-запити обробляються ресурсом OpenFlow, визначеним у додатку mywebserver. Для кожного веб-сервісу, що надається OpenFlow, існує метод, визначений у MyWebServerResource. Концепції, що стосуються кожного компонента додатка WebServer, описані нижче.

Додаток NOX реалізує платформу для розгортання Веб-сайтів, як аплікація NOX. Ця функція використовується для реалізації мережі послуг як особливий вид веб-сайтів. Клас mywebserver - це NOX додаток для реалізації класу. Він запускає всі додатки, які повинні запускатися одночасно з додатком WebServer. Він також запускає NOX за замовчуванням і визначає його ресурс як об'єкт класу MyWebServerResource. Клас MyWebServerResource визначає ресурс, який є своєрідним веб-сайтом, реалізованим стандартом NOX. Цей клас також реалізує відображення URL-запитів для виклику функцій, що забезпечують інтерфейс між користувачем і OpenFlow мережею. Є деякі послуги, вже реалізовані на

MyWebServerResource. До кожної служби можна звернутися за HTTP-запитом, використовуючи певний URL. Послуги описано нижче:

GetStats: ця служба не приймає жодних параметрів. Вона викликає додаток Stats і повертає статистику та інформацію про перемикач OpenFlow мережі в XML-повідомленні.

GetTopology: ця служба не приймає жодних параметрів. Вона викликає Discovery App і повертає топологію мережі в XML-повідомленні. Ця служба повертає список усіх мережевих посилань.

GetNeighbor: ця служба не приймає жодних параметрів. Вона викликає Discovery App і повертає топологію мережі в XML-повідомленні. Ця служба повертає список усіх сусідніх вузлів для кожного вузла мережі.

GetSpanningTree: ця служба не приймає жодних параметрів. Вона викликає App Discovery і повертає сполучне дерево мережі в XML-повідомленні. Ця служба повертає список сусідніх вузлів, які пов'язані з вузлом за сполучним деревом, для кожного вузла в мережі.

AddFlow: ця служба використовує характеристики потоку, такі параметри, як потік, тайм-аут простою, жорсткий тайм-аут, пріоритет і дія. Ця послуга додає новий потік, що викликає додаток FlowManager, який виконує необхідну дію над мережею.

DelFlow: ця служба приймає як параметри характеристики потоку, такі як збіг потоку, тайм-аут простою, жорсткий тайм-аут і пріоритет. Ця служба видаляє потік, що викликає додаток FlowManager, який виконує необхідну дію в мережі.

MigrateFlow: ця служба використовує характеристики потоку, параметрами яких є, наприклад, збіг потоку, тайм-аут простою, жорсткий тайм-аут, пріоритет, дія, а також список, які мають бути встановлені. Ця послуга переносить виклик потоку додатку FlowMigration, який виконує необхідні дії по мережі.

Графічний інтерфейс користувача

OpenFlow перенаправляє мережевий трафік відповідно до таблиці потоків, що містить активні потоки. Ця таблиця містить характеристики і правила потоку, які повинні застосовуватися, наприклад, для визначення черги і вихідного порту.

Ця таблиця може бути налаштована локально або мережевим контролером. Зручний інтерфейс може бути так реалізовано, щоб дозволити користувачам змінювати таблиці потоків, і полегшити конфігурації управлінням мережею. Цей користувальницький інтерфейс був розроблений на основі веб-додатку, в якому, використовуючи веб-браузер, користувач має доступ до інтерфейсу і може запускати команди і запити для управління мережею.

Додаток, який надає графічний інтерфейс, ділиться на три рівні.

Перший рівень - це рівень даних, де виконуються користувацькі команди і де збір даних виконується у вигляді відповідей на запити.

Другий рівень - рівень обробки даних, який обробляє всю отриману інформацію, відправляючи її на інші рівні.

Третій рівень - це рівень презентації, який організовує і показує дані користувачеві.

Ізоляція, що забезпечується шаром структур, дає змогу модифікувати певний шар без зміни інших. На рисунку 2.11 показано рівні додатка та використовувані протоколи для обміну повідомленнями.

Рівень даних: рівень даних складається з контролера NOX і його додатку. Додаток WebServer App відповідає за надання комунікаційного інтерфейсу між шаром даних і шаром обробки даних. Цей додаток взаємодіє з рівнем обробки даних через HTTP-протокол.

Рівень обробки даних: рівень обробки даних складається з Веб сервера, який обробляє запити та команди від користувачів. Проектування власного сервера, замість використання наявного сервера, такого як Apache, може бути корисним. Пропозиція адміністраторам повного контролю над наданими послугами сприяє розвитку системи.

Рівень представлення даних: рівень презентації складається з мітки і скриптів, які інтерпретуються веб-браузером. Вони можуть бути у вигляді HTML, JavaScript (JS), CSS, XML і SVG. HTML-файли мають веб-інтерфейс, який веб-браузер інтерпретує, щоб показати цю веб-сторінку.

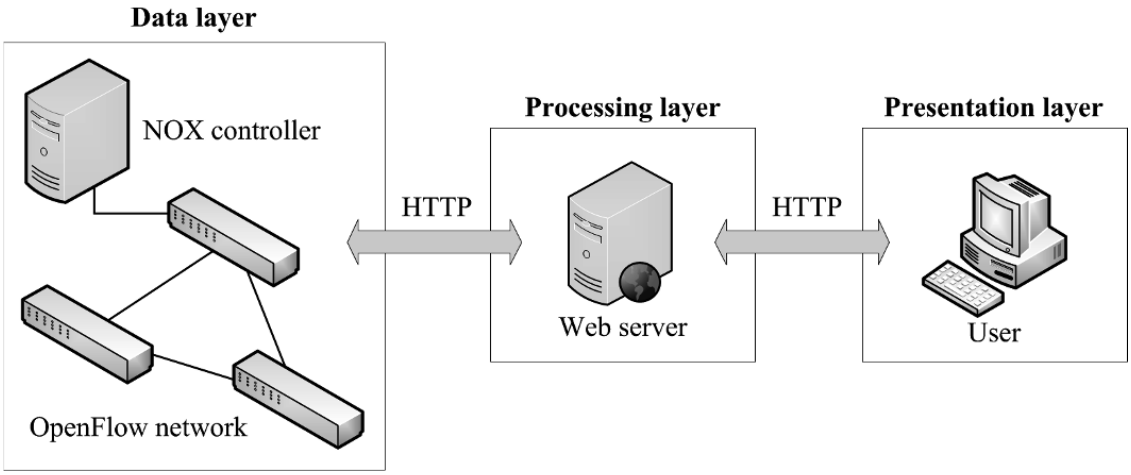


Рисунок 2.11 - Прикладні рівні

CSS-файли мають розмітку стилю, щоб поліпшити вигляд, наданий HTML. Файл CSS інтерпретується веб-браузером і застосовує стиль до розмітки HTML. Файли JS мають функції скрипта для створення динамічних та інтерактивних веб-сторінок. XML-файли мають інформацію про веб-браузер або JS, щоб показувати або обмінюватися даними. Повідомлення, які надані як відповіді на команди користувача, являють собою XML-повідомлення. SVG-файли мають XML-опис топології мережі для надання графічної візуалізації. Комбінація між SVG і JS дає змогу створювати анімацію та сприяє взаємодії із зображеннями, створеними за допомогою SVG. На цьому рівні є ресурси, які дозволяють користувачам отримувати мережеву інформацію та виконувати команди.

РОЗДІЛ 3. ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ТА УПРАВЛІННЯ ЕЛЕМЕНТАМИ ВІРТУАЛЬНОЇ МЕРЕЖІ

3.1 Xen-based прототип

У другому розділі дисертаційної роботи визначено п'ять примітивів (створення, видалення, міграція, моніторинг і налаштування), які інфраструктура віртуалізації мережі повинна забезпечувати для панелі управління, за допомогою якої здійснюється контроль і управління елементами віртуальної мережі. На рисунку 3.1 показано взаємозв'язок між панеллю керування і загальним віртуалізованим мережевим елементом.

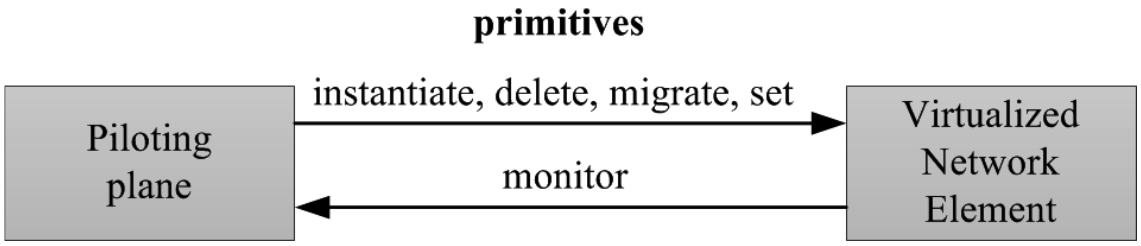


Рисунок 3.1 - Керувальні примітиви та зв'язок з панелі керування і віртуалізованого мережевого елемента

У принципі, панель керування виконує інтелектуальні алгоритми: автоматичне створення/видалення віртуальних мереж, а також міграція мережевих елементів і завдання параметрів розподілу ресурсів. Отже, для панелі керування необхідно отримати інформацію і використовувати примітив монітора. Цей примітив виконує виклики інструментів моніторингу, необхідних для вимірювання змінних, що цікавлять, таких як доступна пропускна здатність, процесор і пам'ять, зв'язок і наскрізна затримка. Після моніторингу, панель управління може діяти в мережі з використанням чотирьох примітивів.

Параметри, доступні для моніторингу та налаштування, охоплюють низькорівневі й апаратні параметри, як-от кількість віртуальних процесорів, що надаються віртуальному маршрутизатору, і пріоритет використання процесора в конфліктному сценарії. Таким чином, панель управління динамічно адаптує

ресурси, виділені для кожної віртуальної мережі, відповідно до поточного стану мережі, кількості користувачів, пріоритету кожної віртуальної мережі, угод про рівень обслуговування (SLA) тощо. Ми вважаємо, що елемент віртуальної мережі має дві основні панелі: панель віртуалізації та управління. Платформа віртуалізації забезпечує субстрат для роботи логічних мережевих елементів в одній загальній фізичній мережі. Панель управління забезпечує основу для оптимізації продуктивності мережі на основі SLA для кожної віртуальної мережі. На рисунку 3.2 показано архітектуру загального елемента віртуальної мережі. Ядром вузла є система віртуалізації, яка може бути реалізована різними інструментами віртуалізації, такими як Xen або OpenFlow. Датчики та приводи також вважаються складовими елементами системи віртуалізації. Датчики збирають інформацію, що дає змогу описати контекст, до якого вони належать, а виконавчі механізми - це програмні компоненти, які застосовують дії, що вимагаються панеллю керування на мережевому рівні. На рисунку 3.2 контролер є панеллю керування, яка приймає й агрегує інформацію, надіслану вузлами, і потім надсилає команди для встановлення параметрів і виконання керівних дій на вузлах. Усі вузли повинні забезпечувати інтерфейс із панеллю керування і мати можливість обмінюватися з ним повідомленнями для керування у форматі Extensible Markup Language (XML). У цьому розділі розглядаються конкретні зміни віртуалізації для реалізації п'яти примітивів, використовуваних адміністраторами, а також для підвищення продуктивності віртуалізованих мережевих елементів. Кожен інструмент віртуалізації вводить інший набір датчиків, а також вимагає різні механізми управління мережею і реалізації примітивів панелі управління. Наприклад, Xen надає вбудовані приводи для створення та знищення віртуальних машини (VM), але власний механізм міграції VM недостатньо адаптований до додатків віртуального маршрутизатора, тому що він не уникає втрати пакетів. Тому має бути забезпечений ефективний механізм міграції. Подібно до Xen, OpenFlow надає інструменти для створення та знищення потоків, але він також не має механізму міграції рідного потоку. Таким чином,

шляхи з впровадження детально описані для двох конкретних прототипів на основі інструментів віртуалізації Xen і OpenFlow.

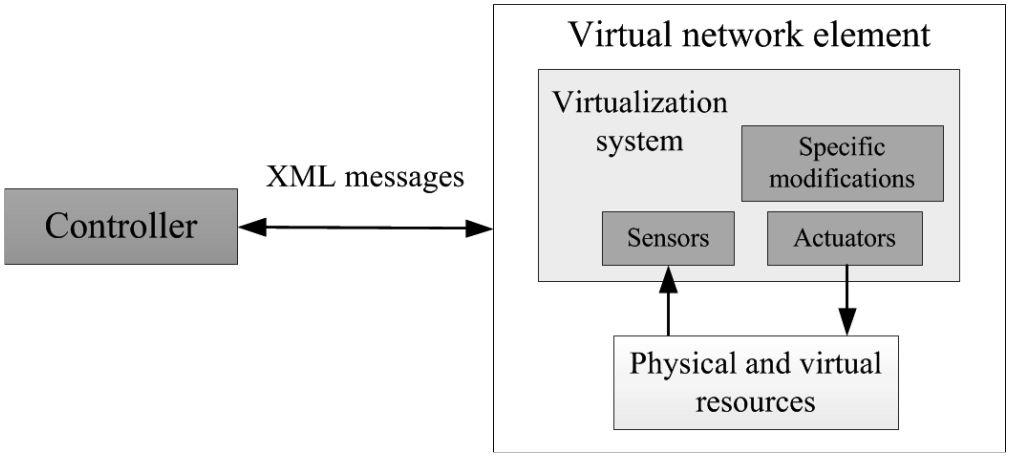


Рисунок 3.2 - Архітектура спільного елемента віртуальної мережі: контролер приймає дані, отримані датчиками, і надсилає команди для управління фізичними та віртуальними ресурсами мережевих елементів

Прототип мережі на основі Xen складається з декількох модулів, які виконуються на різних типах вузлів. У принципі, кожен вузол відіграє якусь роль: контролера, маршрутизатора або ж клієнта. Контролер - це спеціальний вузол, який приймає і консолідує дані, отримані від усіх мережевих вузлів. Вузол контролера отримує дані і відправляє команди фізичним і віртуальним маршрутизаторам. Маршрутизатор вузла - це мережеві субстрати. Один фізичний маршрутизатор запускає один або кілька віртуальних маршрутизаторів. Як фізичні, так і віртуальні маршрутизатори запускають кілька модулів, які контролюють і оперативно діють під час приймання команд. Нарешті, клієнтський вузол дає змогу користувачам взаємодіяти з контролером за допомогою графічного користувацького інтерфейсу (GUI). Цей графічний інтерфейс надає інформацію про прототип для моніторингу користувачів, а також простий інтерфейс керування. На рис. 3.3. наведено основні модулі Xen-прототипу та його інтерфейси. Контролер має два модулі: сервера віртуальної машини (VMS) і зв'язку з клієнтом. VMS є ядром контролера. Який має інтерфейс протоколу простого доступу до об'єктів (SOAP) для взаємодії як з панеллю керування, так і з клієнтським вузлом. VMS об'єднує всі види інформації та

виконує всі алгоритми керування та обслуговування. Клієнт комунікаційного модуля використовується VMS-ом для зв'язку з іншими модулями маршрутизатора.

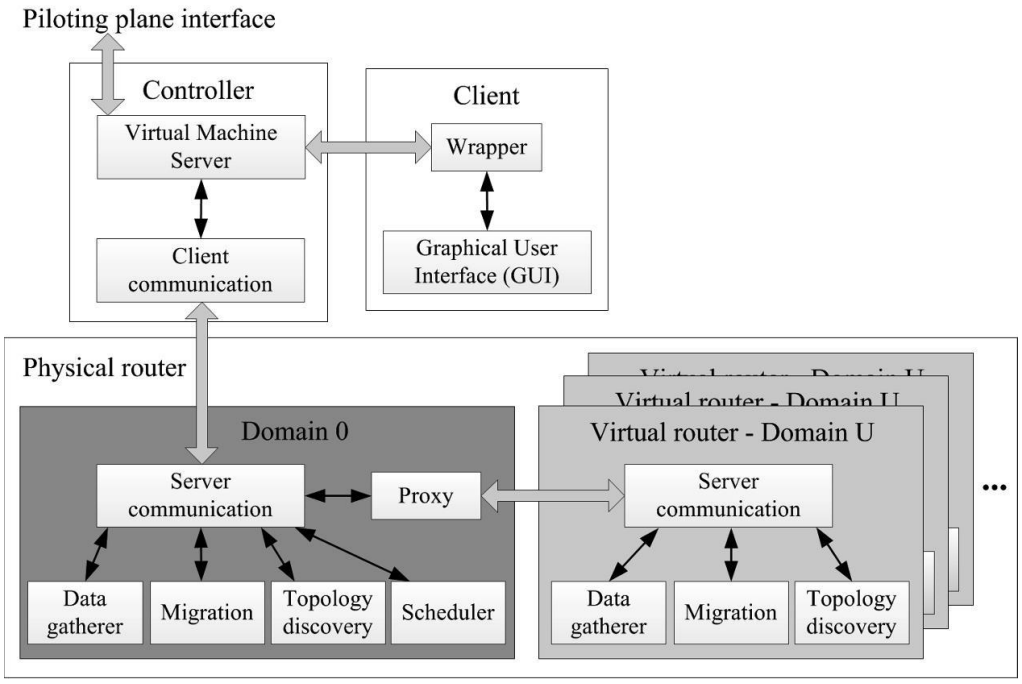


Рисунок 3.3 - Архітектура прототипу Xen: Основні модулі та відносини

Фізичні маршрутизатори забезпечують базу, використовувану віртуальними мережами. Кожен фізичний маршрутизатор виконує модуль зв'язку із сервером, який приймає запити від контролера і перенаправляє ці запити на певні модулі. Якщо запит адресований даному віртуальному маршрутизатору, він перенаправляється на конкретний віртуальний маршрутизатор за допомогою модуля Proxy. В іншому разі його відправляють до одного з інших модулів, що працюють на фізичному маршрутизаторі. Наприклад, якщо отриманий запит розглядає завдання моніторингу, його пересилають у Data Gatherer модуль для отримання параметрів із фізичних і віртуальних маршрутизаторів. Якщо отриманий запит відповідає дії, пов'язаній зі спільним використанням ресурсів серед віртуальних маршрутизаторів, модуль Scheduler обробляє цей запит. Модуль Topology Discovery обробляє запити, пов'язані з фізичним або віртуальним виявленням топологій мереж. Запити на міграцію обробляються модулями міграції, які забезпечують міграцію віртуального маршрутизатора без

втрати пакетів. Клієнтський вузол дає змогу користувачам моніторити і контролювати прототип мережі. Цей вузол має графічний інтерфейс, який показує фізичні та віртуальні мережеві топології, а також додатково дає змогу контролювати дрібний матеріал, демонструючи детальну інформацію про обраний фізичний або віртуальний маршрутизатор. Клієнтський вузол також дає змогу користувачам надсилати команди прототипу. Наприклад, користувачі можуть виконати міграцію віртуального маршрутизатора з одного фізичного маршрутизатора на інший фізичний маршрутизатор за допомогою кліка комп'ютерною мишкою. GUI, запущений на клієнті, взаємодіє з контролером за допомогою модуля Wrapper. Цей модуль перетворює команди з GUI на SOAP, а потім викликає контролер.

Міграція - важливий контрольний примітив, оскільки дає змогу панелі управління динамічно перебудовувати логічну топологію мережі, без порушення роботи служб і без втрати пакетів. Xen має власний механізм міграції, розроблений для переміщення віртуальних машин. Цей механізм заснований на двох припущеннях:

- міграція відбувається всередині локальної мережі;
- жорсткий диск VM спільно використовується в мережі.

Таким чином, міграція віртуальної машини складається з двох основних процедур:

- копія віртуальної машинної пам'яті до нової фізичної локації;
- реконфігурація мережі посилання без розриву з'єднань.

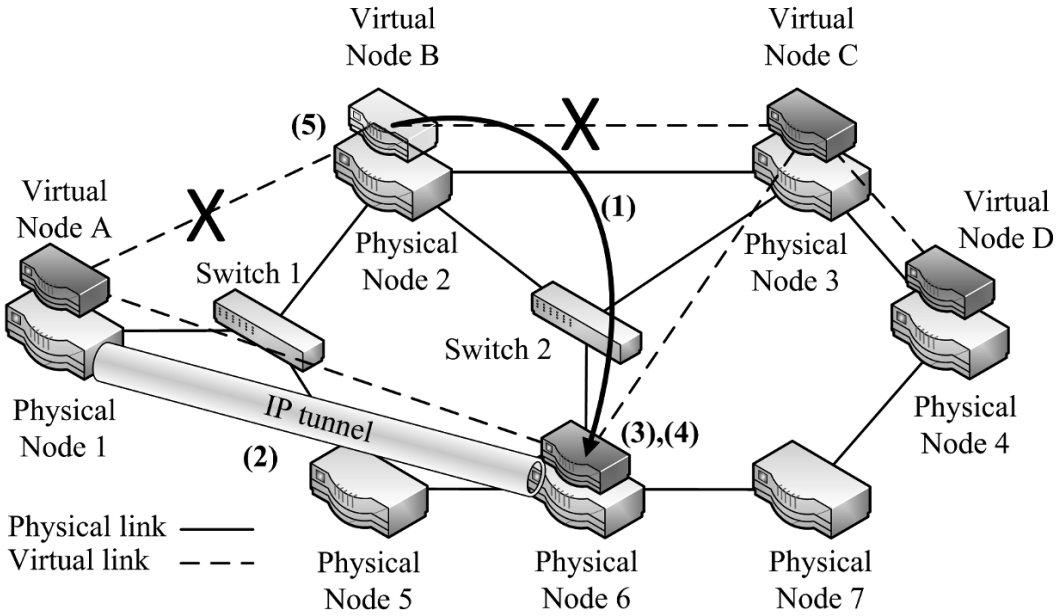
Існує кілька варіантів виконання копії пам'яті. Найпростішою альтернативою є призупинення віртуальної машини, перенесення всіх блоків пам'яті на новий фізичний вузол, а потім відновлення роботи віртуальної машини. Незважаючи на простоту, ця альтернатива страждає високим часом простою, тобто час, коли віртуальна машина недоступна під час міграції. Щоб скоротити час простою, процедура призупинення-передачі-відновлення pre-copy migration, яка складається з двох етапів:

- ітеративне попереднє копіювання;

- копіювання.

На першому етапі всі сторінки пам'яті переносяться на нову фізичну машину, за винятком тих, які називаються «гарячими сторінками», що є найбільш часто зміненими сторінками. Таким чином, час простою скорочується, оскільки тільки кілька сторінок, а саме гарячі сторінки, передаються, поки віртуальна машина вимкнена. Власний механізм VM-міграції Xen добре працює для додатків консолідації серверів, але він не ефективний для міграції віртуального маршрутизатора. Через використання механізму попередньої копії, час простою перебуває в порядку сотні мілісекунд, і за цей час пакети втрачаються. Залежно від швидкості передачі даних, віртуальні маршрутизатори зазнають втрати великої кількості пакетів під час простою. Також важливо мінімізувати загальний час міграції, щоб гарантувати, що ми можемо швидко звільнити ресурси фізичної машини. Ще одна проблема власного механізму Xen для перенесення віртуальних маршрутизаторів - це те, що він припускає, що міграція завжди виконується з одного фізичного маршрутизатора до іншого фізичного маршрутизатора в тій же локальній мережі (LAN). В Інтернеті, ми не можемо припустити, що фізичні вузли завжди належать до однієї і тієї ж локальної мережі. Щоб перенести віртуальні маршрутизатори без втрати пакетів, альтернативою є реалізація методу поділу площин у Xen. Механізм міграції з площинною сепарацією гарантує відсутність втрати пакетів у площині даних під час міграції віртуальної машини. Також відсутня втрата пакетів управління. Механізм вставляє тільки затримку доставки пакета управління. Модифікований механізм, однак, заснований на міграції за замовчуванням Xen, тобто він також вимагає, щоб маршрутизатори перебували в одній локальній мережі. Крім того, відображення віртуального зв'язку за кількома фізичними посиланнями залишається відкритою проблемою, яка залежить від тунелі Інтернет-протоколу (IP) або створення нових віртуальних маршрутизаторів у мережі. Наприклад, на рисунку 3.4 ми переносимо Віртуальний Вузол від фізичного вузла 2 до фізичного вузла 6. Фізичний вузол 6, однак, не є сусідом із фізичним вузлом 1. Отже, для завершення ми повинні створити тунель від фізичного вузла 6 до фізичного вузла 1, для імітації околиці з

одним хопом. Інше рішення створити новий віртуальний роутер для заміни тунелю. Це рішення, однак, змінює віртуальну топологію і впливає на роботу протоколу маршрутизації. Є також інший компонент, який використовується під час процесу міграції, Міграція потрібного компонента. Цей компонент працює як за джерелом, так і за призначенням і повідомляє про успіх або провал процесу міграції.



(1) Migrate control plane from Physical Node 2 to 6 maintaining data and control planes
(2) Do interface dynamic binding and create tunnel between Physical Nodes 1 and 6
(3) Create a new forwarding table in Domain 0 of Physical Node 6
(4) Reconfigure links with ARP reply
(5) Delete data plane on Physical Node 2

Рисунок 3.4 - Приклад міграції маршрутизатора на основі Xen, коли віртуальне посилання відображається в багато інтервальний шлях у фізичній мережі

Xen-топологія. Модуль топології виявляє топології як фізичних, так і віртуальних мереж. Цей модуль перевіряє всіх сусідів кожного мережевого елемента, використовуючи Nmap Security Scanner. Модуль топології має три компоненти: сусіди, що сканують, консолідація топології та вузол об'єднання.

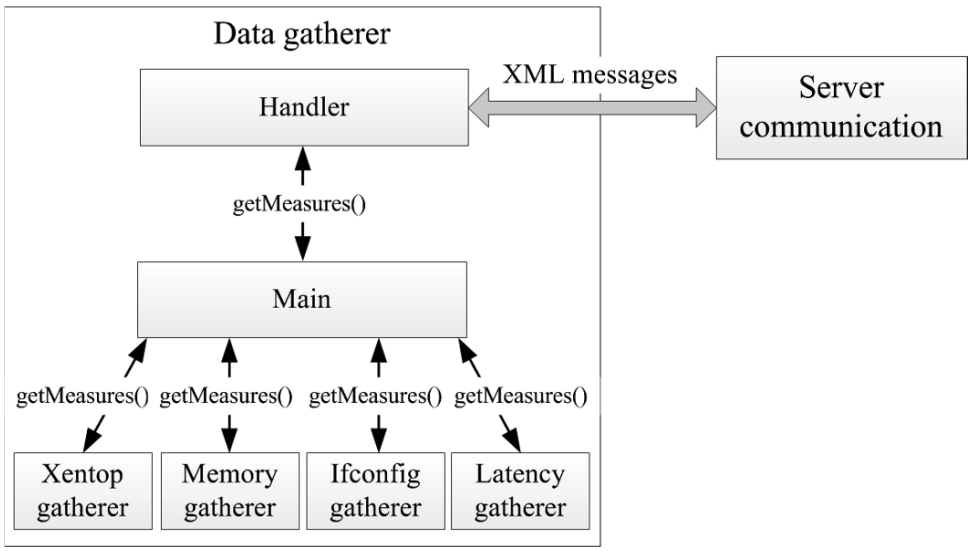


Рисунок 3.5 - Архітектура Data Gatherer: основний модуль надсилає запити на конкретні модулі моніторингу, а потім пересилає отримані дані обробнику, який обробляє ці дані

Компонент сканування відповідає за виявлення всіх сусідів певного мережевого елемента. Виконуються як фізичні, так і віртуальні мережеві елементи цього компонента. Пошук сусіда використовує інструмент Nmap, який перевіряє весь діапазон, IP кожного мережевого інтерфейсу, елементів мережі. Ми додаємо до списку адрес усі IP-адреси, які відповідають стандартам. Після цього у нас є всі сусіди мережевого елемента, які пов'язані всіма мережевими інтерфейсами. Інформація про сусідство - це володіння інформацією про IP-адресу і MAC-адресу сусіда, і про латентність посилання. Після виявлення сусідів усіх мережевих інтерфейсів, компонент створює список оточення і передає його компоненту вузла консолідації через XML-повідомлення, як на рис. 3.6. Компонент «Вузол Консолідації» виконується на кожному фізичному елементі мережі. Цей компонент спрямований на отримання інформації про сусідів цього елемента як у фізичних, так і у віртуальних мережах. Топологія консолідації компонента викликає компонент «Консолідація вузла», який має три завдання. Перше завдання - виявити фізичних сусідів елемента мережі. Друге завдання - виявити всіх сусідів кожного елемента віртуальної мережі, які працюють над цим фізичним мережевим елементом. І ті й інші процедури виявлення використовують

компонент «Сканування». Перші два завдання виконуються в один і той самий час, тому що між ними немає залежності. Останнє завдання компонент «Консолідація вузла» призначений для консолідації інформації. Консолідація складається з інформації про сусідство фізичного елемента мережі та списку елементів віртуальної мережі зі своїми сусідами. Модуль повертає консолідовану інформацію контролеру, який запускає топологію консолідації компонента з використанням повідомлень у форматі XML.

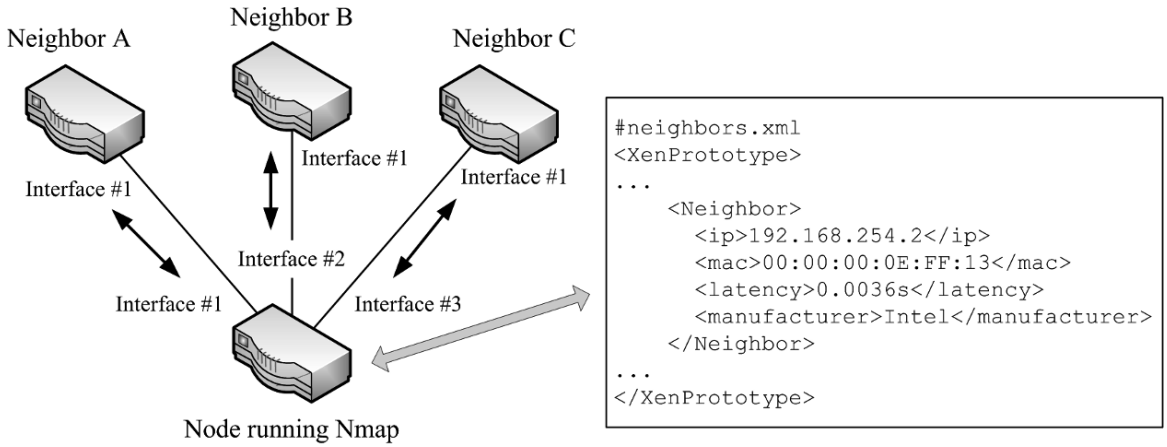


Рисунок 3.6 - Компонент «Сусіди, що сканують»: інструмент nmap використовується для зондування всіх інтерфейсів мережевого елемента, а отже, вузли, які реагують на ці зонди, додаються до списку околиць

VMS має список зареєстрованих вузлів фізичної мережі, але немає інформації про взаємозв'язки між цими вузлами. Модуль топології надає інформацію про підключення. Цей модуль запитує кожен фізичний вузол у списку зареєстрованих вузлів про інформацію про з'єднання, про фізичні та віртуальні елементи. Після отримання всієї інформації про з'єднання, компонент топології консолідації обчислює топологію кожної мережі. Ми моделюємо топологію мережі як графік. Тому компонент повертає графік подання у VMS. Графік представлений матрицею суміжності кожної мережі. Таким чином, модуль топології дає змогу VMS надавати поточну топологію фізичних і віртуальних мереж.

3.2 OpenFlow прототип

Прототип OpenFlow заснований на наборі додатків, запущених над NOX контролером. NOX - це контролер OpenFlow, який налаштовує та керує потоками в комутаторах OpenFlow. Розроблений додаток NOX реалізує п'ять необхідних примітивів: екземпляр потоку (екземпляр примітиву), видалення потоку (видалити примітив), міграція потоку (мігрувати примітив), управління та зміна потоків (набір примітивів), моніторинг мережі та виявлення топології (примітив монітора), а також функції, щоб уникнути циклів у мережі. Існує також інтерфейс веб-сервісу для надання п'яти примітивів системі управління. Архітектура прототипу OpenFlow проілюстрована на рисунку 3.7. Прототип виконує роль як датчиків, так і виконавчих механізмів, що визначені примітивами системи управління. Приводи являють собою диспетчерів потоків і потоків додатків для міграції. Додаток Flow Manager пропонує створювати екземпляри / видаляти / встановлювати примітиви. Додаток Flow Manager реалізує інтерфейс між іншими додатками NOX і командами OpenFlow. Цей додаток відповідає за додавання, зміну та видалення потоків. Додаток Flow Manager отримує запит на операцію потоку і переводить його в команду OpenFlow. Іншим виконавчим механізмом є міграція потоку додатку, який реалізує примітив міграції. Міграція потоку переносить потік від одного шляху до іншого без втрати пакетів. Статистика і додатки виявлення впроваджують датчики в прототипі. Обидва додатки задовольняють примітив монітора. Додаток «Статистика» вимірює мережі та збирає статистику про комутатори. Додаток Discovery виявляє мережеву топологію і будує графік для представлення мережі. Існує також додаток Spanning Tree, який дає змогу уникнути появи петлі в мережі та непотрібні повторні передачі в ефір.

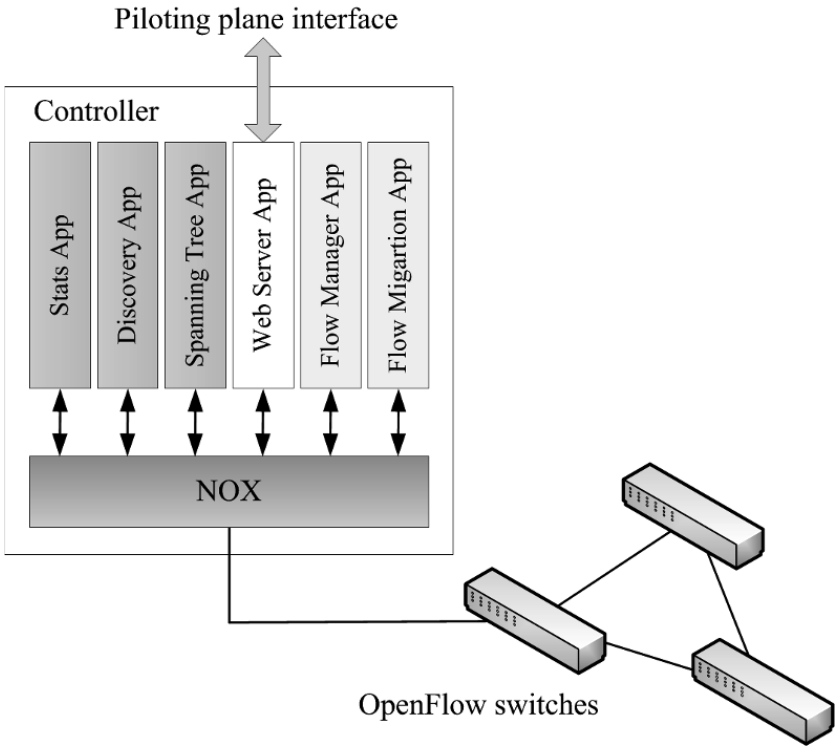


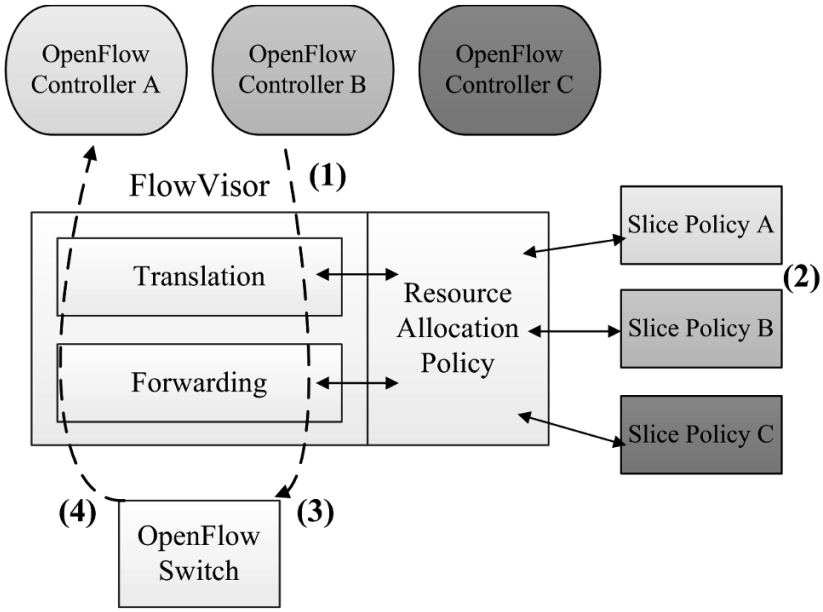
Рисунок 3.7 - Архітектура прототипу на основі OpenFlow

Крім того, використовується інструмент, який дає змогу кільком контролерам NOX працювати паралельно для того, щоб розділити мережу на кілька віртуальних. Цей інструмент, який називається FlowVisor, керує спільним використанням мережевих ресурсів, таких як пропускна здатність, топологія, спостереження за контролерами, трафік кожної активності, перемикає використання ЦП і таблицю потоків. FlowVisor реалізує заданий примітив для налаштування параметрів кожної віртуальної мережі.

FlowVisor. FlowVisor є спеціальним типом контролера OpenFlow. Він працює як прозорий проксі між мережевими пристроями та іншими контролерами, такими як контролери NOX. Основною особливістю FlowVisor є можливість розділяти мережі та обмінюватися мережевими ресурсами контрольованими та ізольованими чином. Як показано на рисунку 3.8 (адаптовано з Sherwood et al.), FlowVisor перехоплює повідомлення OpenFlow, відправлені гостьовими контролерами. Після FlowVisor, заснований на політиці відокремлення користувачів (2), прозоро змінює повідомлення, щоб розділити елемент управління по різних мережах. Тільки FlowVisor пересилає повідомлення

від комутаторів до гостей, якщо повідомлення відповідають заданій політиці. FlowVisor розділяє мережу, зберігаючи розділи, ізольовані один від одного.

FlowVisor може віртуалізувати мережу, розділяючи ресурси комутаторів серед кількох контролерів. Крім того, FlowVisor дає змогу створювати ієрархії FlowVisor - ів, що змінюють мережеву архітектуру, як показано на рисунку 3.9. У цьому разі один екземпляр FlowVisor (FlowVisor 2) віртуалізує кілька комутаторів паралельно (Перемикачі з 1 по 4), а FlowVisor 1 рекурсивно розділяє віртуальні розділи, визначені FlowVisors 2 і 3. Щоб розділити мережу між контролерами, FlowVisor фокусується на спільному використанні п'яти мережевих ресурсів: ізоляція смуги пропускання, топологія виявлення, управління трафіком, таблиці моніторингу та контролю пересилання процесорів пристроїв.



FlowVisor:
(1) Intercepts the OpenFlow messages sent by guest controllers.
(2) Verifies the user slicing policy.
(3) Modifies the message to delimit the control to a slice of the network.
(4) Forwards messages from switches to guests, if the messages match the slice policy.

Рисунок 3.8 - Операції, що виконуються FlowVisor

Наприклад, ізоляція смуги пропускання забезпечується за рахунок використання різних черг пріоритетів на комутаторах. Одна черга ізольована від інших. Усі пакети в потоці позначені одним і тим самим ідентифікатором, а потім відображаються на одну з восьми черг пріоритетів. Потоки можна класифікувати

на основі трьох різних ідентифікаторів: точка пріоритету VLAN (PCP), тип IP Сервісу (ToS) і якість обслуговування OpenFlow (QoS). За замовчуванням використовується VLAN PCP. Важливо зазначити, що гарантовано мінімальна пропускна здатність.

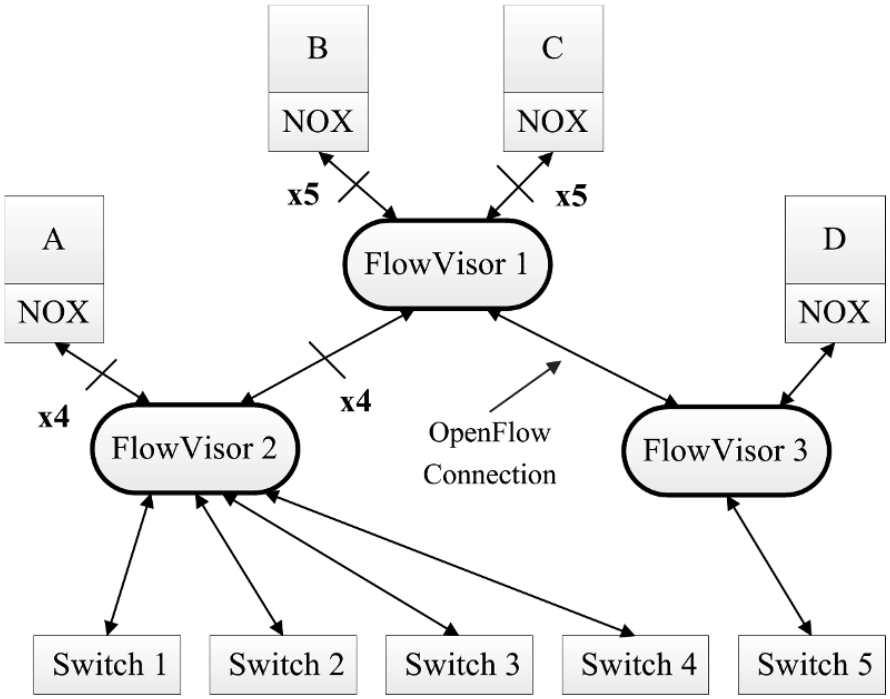


Рисунок 3.9 - Ієрархія FlowVisor: FlowVisor 1 рекурсивно розділяє віртуальні розділи, визначені FlowVisors 2 і 3

Міграція OpenFlow. Міграція потоку - це додаток NOX, розроблений для прототипу, який визначає новий шлях між джерелом і пунктом призначення OpenFlow, який перемикає і змінює поточний шлях потоку на новий шлях без втрат пакетів, під час цієї процедури. Щоб перенести потік, міграція потоку додатка приймає в якості параметрів список комутаторів. Цей список визначає впорядковану групу перемикачів, через які повинен пройти новий потік у шляху. Обчислювати найкоротший шлях від джерела до пункту призначення, включно з обраними комутаторами, він використовує узагальнення алгоритму Дейкстри. Додаток Flow Migration має два інтерфейси: один із яких є серверним додатком із веб-інтерфейсом, за допомогою якого змінюються параметри керування, і комутатор, де додаток відправляє конфігурацію потоку команди. Рисунок 3.10 ілюструє приклад міграції потоку в процесі. Спочатку джерело відправляє пакети

в пункт призначення через Шлях 1, складений вузлами A-F-E-D. Новий потік від комутатора від A до D з вимогою, щоб він проходив через комутатори A, B і D, темно-сірі вузли A, B і D не є повним шляхом від A до D. Таким чином, додаток Flow Migration обчислює повний шлях від ключів джерела до адресата, використовуючи алгоритм Dijkstra, щоб мінімізувати кількість переходів на новому шляху. Згідно з алгоритмом, вузол C має бути включений у шлях. Після визначення всіх вузлів, що складають весь шлях, додаток налаштовує потік у перемикачах у зворотному напрямку, тобто він налаштовує потік від найближчого комутатора до місця призначення, до найвіддаленішого перемикача. Оскільки всі шляхи від місця призначення до джерела вже налаштовані, але посилання вихідного комп'ютера на комутатор A, яке може бути змінено, дозволяє уникнути втрат пакетів.

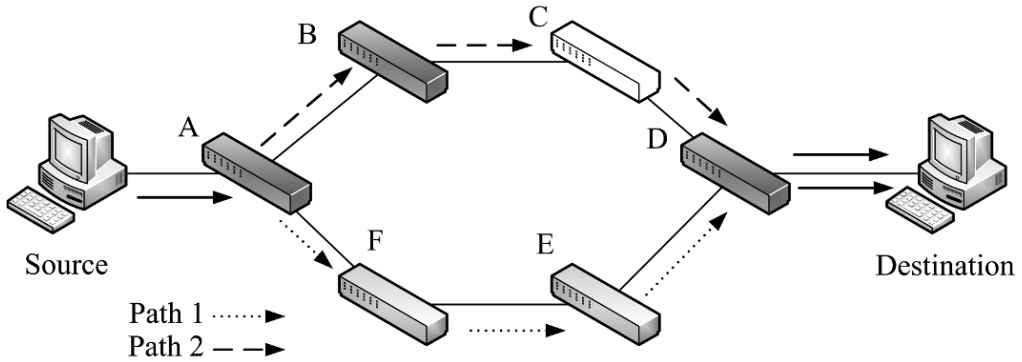


Рисунок 3.10 - Робота програми перенесення потоку

Відкриття OpenFlow. Додаток Discovery являє собою модифіковану версію стандартної NOX заяви. По-перше, цей додаток сумісний з будь-яким OpenFlow, і також розроблено інтерфейс для веб-сервера. Додаток Discovery також отримує топологію як фізичних, так і віртуальних мереж. Здебільшого додаток Discovery реалізує виявлення Link Layer Discovery Протокол (LLDP). Цей протокол дає змогу вузлам передавати інформацію про можливості та поточний стан мережевих пристроїв. Реалізація LLDP є необов'язковою в стеку протоколів станцій IEEE 802 LAN. Формат релізу LLDP показано на рисунку 3.11. LLDP реліз має інформацію про пристрій, що зберігається в TLV структурі. Потім додаток Discovery створює LLDP-фрейм для всіх портів цього комутатора.

Комутатор передає ці релізи через свої порти. Після отримання першого релізу LLDP комутатор не знає правил пересилання для цього фрейму. Перемикач переводить фрейм контролеру для аналізу його вмісту. Після отримання релізу, контролер запитує виконуваний екземпляр додатку Discovery для обробки цього фрейму. Таким чином, додаток аналізує, який комутатор отримав реліз, з якого порту був отриманий фрейм, і через який порт був відправлений реліз. За допомогою цієї інформації додаток створює дані структури з перемикачем джерела, портом джерела, перемикачем призначення та призначенням порту. Ця структура даних ідентифікує посилання та агрегує всі посилання мережі, що характеризує топологію мережі.

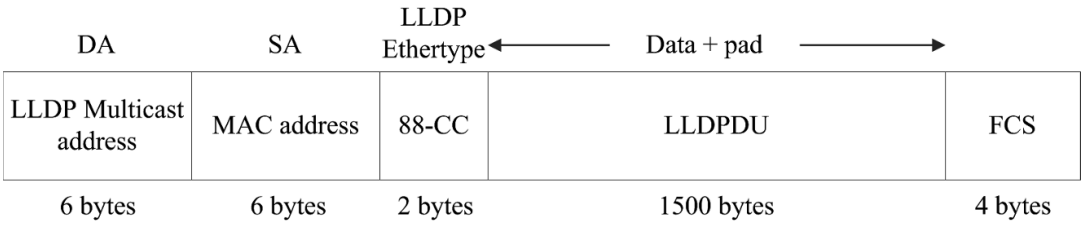
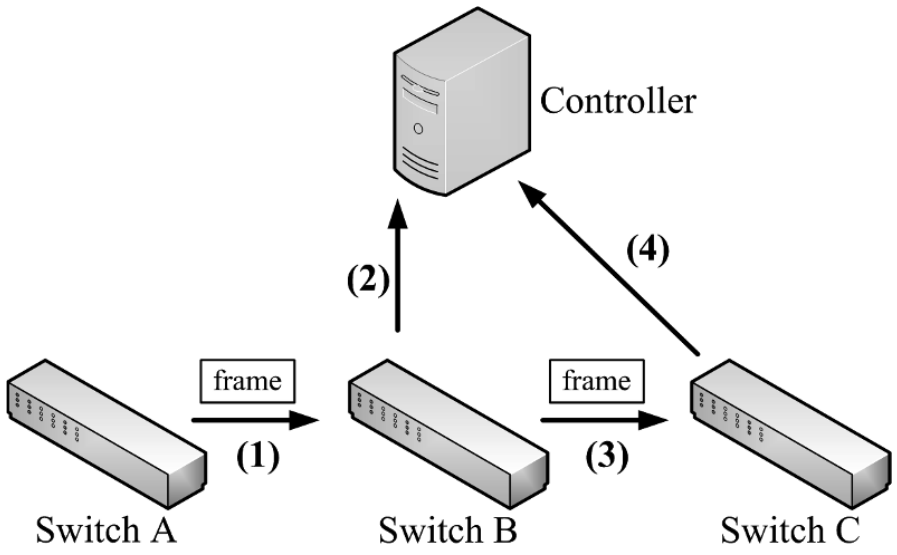


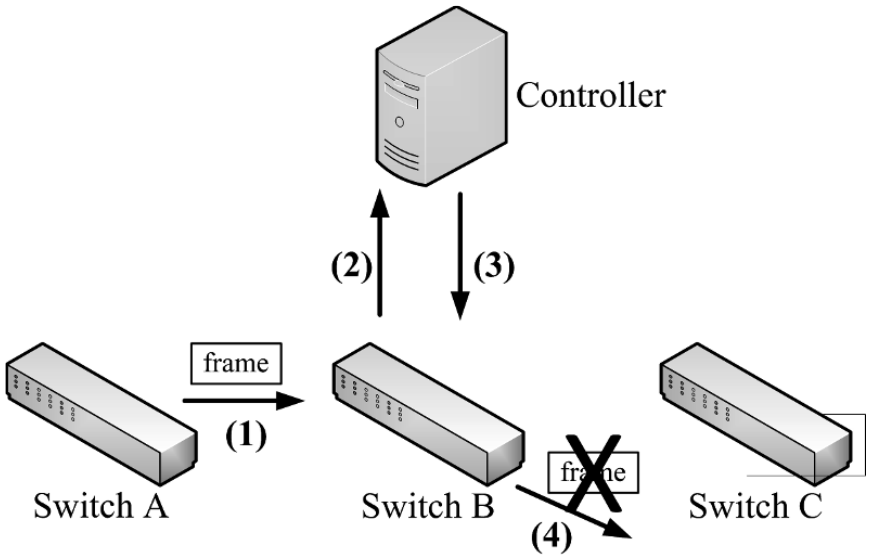
Рисунок 3.11 - Формат кадру LLDP для IEEE 802.3

Алгоритм додатку Discovery NOX за замовчуванням змінено на такий, що гарантує правильність процедури виявлення топології на типі-1, серед перемикачів OpenFlow. У комутаторах типу 0 найпростіша модель OpenFlow комутатора, фрейм пересилається тільки в тому разі, якщо контролер уже налаштував потік для цього фрейма. Якщо немає налаштованого потоку, фрейм буде скинуто. Однак комутатори Type-1 обробляють трафік потоку, навіть якщо немає специфічного правила, налаштованого контролером, тобто за замовчуванням вихідні пакети не відкидаються автоматично. Цей факт призводить до неправильного уявлення топології мережі. Припустимо, що LLDP відправлено з комутатора A на комутатор B. З налаштуванням за замовчуванням алгоритму, контролер не відправить команду, щоб видалити цей фрейм після його обробки. У цьому випадку B може неправильно переспрямувати цей фрейм LLDP на інші перемикачі, під'єднані до нього. Припустимо, що комутатори B і C безпосередньо під'єднані, але C не пов'язане безпосередньо з A. У цьому разі B

переводить LLDP до C, отже контролер помилково ідентифікує зв'язок між A і C. Як наслідок, надається неправильна топологія. Щоб усунути цю проблему, ми змінимо алгоритм за замовчуванням, щоб дозволити йому завжди відправляти частини команди до перемикачів. Відмінності між двома алгоритмами представлено на рисунку 3.12.



а) Алгоритм за замовчуванням



б) модифікований алгоритм

Рисунок 3.12 - Приклад поведінки за замовчуванням і зміненого алгоритму виявлення: перші два кроки обох механізмів однакові, але модифікований алгоритм, контролера відправляє команду drop на Switch B, яка скидається

Рисунок 3.12 a) представляє алгоритм виявлення за замовчуванням. Усі перемикачі Типу 1. У цьому прикладі (1) комутатор A надсилає LLDP-кадр для перемикання B, а після, отримуючи цей кадр, (2) B надсилає його на контролер. Потім контролер вказує, що існує зв'язок між A і B. Контролер не (3) переводить фрейм LLDP у C. Перемикач C отримує фрейм, і, отже, (4) відправляє його на контролер. Потім контролер неправильно припускає, що існує зв'язок між A і C. Модифікована версія алгоритму проілюстрована на рисунку 3.12b). Перші два кроки такі самі, як і з алгоритмом за замовчуванням. Різниця в тому, що після цього, (3) контролер відправляє команду скидання в B і, отже, (4) B відкидає LLDP замість пересилання на C. Додаток за замовчуванням надає тільки фізичну топологію мережі, і нам також потрібна інформація про віртуальні мережі в прототипі. Для досягнення цієї мети, модифікований додаток Discovery отримує визначення віртуальної мережі, наприклад ідентифікатор VLAN або діапазон IP, а потім він надає структуру з топологією віртуальної мережі, яка містить усі комутатори, що перенаправляють трафік для цієї віртуальної мережі.

ВИСНОВКИ

Технології віртуалізації як віртуальні приватні мережі та віртуальні локальні мережі часто використовуються спільно. І підходи до розгортання віртуальних мереж не завжди однакові, часом цей вибір іноді навіть не залишається під час вибору фізичного обладнання, придбаного від деяких виробників, які підтримують вкрай вузький список прототипів, підходів віртуальних мереж. Віртуальні мережі так само відрізняються своїми топологіями, і при комбінуванні багатьох показників, таких як і топологія мережі, як прототип мережі, технології, заздалегідь визначені правила адресації в мережі, тощо, які широко впливають на продуктивність даної мережі. А в роль продуктивності мережі виступає її пропускна здатність.

У даній роботі під час дослідження виявилося, що в різних результатах кожен прототип показує хороші результати в тих чи інших тестах, як приклад, Xen прототип вирізнився результатами під час тестування, який був кращим не тільки за OpenFlow, але й за VMWare та ОС на апаратному ресурсі, без використання технологій віртуалізації. Але при цьому OpenFlow вирізнявся відмінністю в методі передачі і з розширеними можливостями. Обидва прототипи мають свої утиліти для створення і управління віртуальним середовищем, а також для їх міграції, контролю і встановлення. Які є основними для обох прототипів, але так само в прототипі Xen є модуль Xentop Gatherer, який збирає дані мережі, а так само Ifconfig Gatherer, The Latency Gatherer і Memory Gatherer. Ці модулі є датчиками, які використовуються в Xen прототипі. Існує також модуль топологій, який дає змогу визначати віртуальні та фізичні мережі. Ці утиліти є привілеями прототипу Xen.

Але в прототипі OpenFlow, так само існують NOX додатки, які є привілеями даного прототипу. Як приклад можна навести додаток «Статистика», який надає інформацію про всі віртуальні комутатори, як-от стан потоків, кількість переданих пакетів тощо. Додаток «Виявлення» виявляє топологію мережі. Але

головною перевагою OpenFlow є додаток «Spanning tree», що дає змогу уникати непотрібного широкомовного трафіку в мережі, непотрібного повторного передавання пакетів і мережевих петель передавання. В обох прототипах підтримується динамічна міграція без втрати пакетів і динамічного реконфігурування мережевих топологій.

Обидва прототипи мають підтримку веб інтерфейсу для зручної конфігурації та управління віртуальними мережами. Додаткові сервіси можуть бути додані в платформу віртуалізації.

Будь-яка мережа вважається ефективною, коли вона оптимально та ефективно виконує свої завдання, поставлені тим чи іншим підприємством. Досягнення ефективності можна досягти повністю проаналізувавши вимоги в мережі, і використавши той прототип, який повністю відповідає вимогам завдання. Але так і слід розглянути всі можливості прототипів і максимально використати їх для створення оптимізованої віртуальної мережі з відповідним набором можливостей.

Дипломна розкриває технології встановлення віртуальних мереж. Два основні прототипи (Xen та OpenFlow), що використовуються для створення віртуальних мереж, були протестовані та випробувані з різним програмним забезпеченням та отримані результати. Було досліджено інтерфейси, показники та програмну підтримку кожного прототипу, і в результаті дослідження було виявлено переваги над ними. В результаті цих досліджень майже всі прототипи віртуалізації мають власне програмне забезпечення, яке використовується в різних напрямках, але результати тестування прототипу Xen перевершили інші прототипи. Отже, обидва прототипи мають свої переваги та недоліки, і їх використання повинно базуватися на вимогах мережі. Виходячи з результатів дослідження, прототип та технології, які відповідають цим вимогам, можуть бути впроваджені адміністраторами мережі. А також можуть бути використані для більш ефективного управління цими віртуальними мережами та оптимізації роботи мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kim-Khoa Nguyen; Cheriet, M.; Lemay, M., "Enabling infrastructure as a service (IaaS) on IP networks: from distributed to virtualized control plane," Communications Magazine, IEEE , vol.51, no.1, pp.136,144, January 2023.
2. YAZICI, Volkan; SUNAY, M. Oguz; ERCAN, Ali O. Controlling a SoftwareDefined Network via Distributed Controllers. arXiv preprint arXiv:1401.7651, 2014.
3. F. Botero, X. Hesselbach, M. Molina, J.R. Amazonas, A novel paths algebrabased strategy to flexibly solve the link mapping stage of VNE problems, January 2013
4. W. P. Herman, Formulação algébrica para a modelagem de algoritmos de roteamento multi-restritivo hop-by-hop, Phd thesis, Escola Politécnica da USP, São Paulo (2008).
5. Jian, Zhang, Li Xiaoyong, and Guan Haibing. "The optimization of Xen network virtualization." Computer Science and Software Engineering, 2008 International Conference on. Vol. 3. IEEE, 2018.
6. Kusnetzky, Dan. Virtualization: A Manager's Guide. " O'Reilly Media, Inc.", 2011.
7. Rhoton, John. Cloud computing explained. Vol. 10. Breinigsville, PA USA: Recursive Press, 2010.
8. Pages, A.; Perelló, J.; Spadaro, S., "Virtual network embedding in optical infrastructures," Transparent Optical Networks (ICTON), 2012 14th International Conference on, vol., no., pp.1,4, 2-5 July 2012
9. Jorge Carapinha and Javier Jim\&\#233;nez. 2009. Network virtualization: a view from the bottom. In Proceedings of the 1st ACM workshop on Virtualize infrastructure systems and architectures (VISA '09). ACM, New York, NY, USA, 73-80. http://doi.acm.org/10.1145/1592648.1592660
10. Azodolmolky, Siamak, Philipp Wieder, and Ramin Yahyapour. "Cloud computing networking: challenges and opportunities for innovations."

Communications Magazine, IEEE 51.7 (2013).

11. Haider, Aun, Richard Potter, and Akihiro Nakao. "Challenges in resource allocation in network virtualization." 20th ITC Specialist Seminar. Vol. 18. 2009.

12. Chowdhury, N.M.M.K.; Boutaba, R., "Network virtualization: state of the art and research challenges," Communications Magazine, IEEE , vol.47, no.7, pp.20,26, July 2009 doi: 10.1109/MCOM.2009.5183468

13. Fischer, A; Botero, J.F.; Till Beck, M.; de Meer, H.; Hesselbach, X., "Virtual Network Embedding: A Survey," Communications Surveys & Tutorials, IEEE , vol.15, no.4, pp.1888,1906, Fourth Quarter 2013

14. Qiang Duan, "Network-as-a-Service in Software-Defined Networks for end-toend QoS provisioning," Wireless and Optical Communication Conference (WOCC), 2014 23rd , vol., no., pp.1,5, 9-10 May 2018

15. J. Inführ and G. R. Raidl, “Introducing the virtual network mapping problem with delay, routing and location constraints,” in Proceedings of the 5th international conference on Network optimization, ser. INOC’11. Berlin, Heidelberg: SpringerVerlag, 2011, pp. 105–117. Pages,

16. Bhardwaj, Sushil, Leena Jain, and Sandeep Jain. "Cloud computing: A study of infrastructure as a service (IAAS)." International Journal of engineering and information Technology 2.1 (2010): 60-63.

17. Ali, Mufajjul. "Green cloud on the horizon." In Cloud Computing, pp. 451-459. Springer Berlin Heidelberg, 2009.

18.Network Virtualization strategy based on Paths Algebra to implement the concept of Infrastructure as a Service 92

18. A; Perelló, J.; Spadaro, S., "Virtual network embedding in optical infrastructures," Transparent Optical Networks (ICTON), 2012 14th International Conference on, vol.,no., pp.1,4, 2-5 July 2012 doi: 10.1109/ICTON.2012.6253704

19. Y. Zhu, M. Ammar, Algorithms for assigning substrate network resources t virtual network components, in: Proc. IEEE INFOCOM, 2006, pp. 2812–2823.

20. J. Lu, J. Turner, Efficient mapping of virtual networks onto a shared substrate, Washington University in St. Louis, Tech. Rep.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ