Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету )
Кафедра кібербезпеки
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: Дослідження механізму ізоляції пам'яті для підвищення безпеки віртуалізованих мережевих середовищах "

Виконав: студент VI курсу, групи СБм-61

Спеціальності:
125 Кібербезпека та захист інформації
(шифр і назва напряму підготовки, спеціальності)
Мазурчак Дмитро Сергійович
підпис (прізвище та ініціали)

Керівник Карпінський М. П.
підпис (прізвище та ініціали)

Нормоконтроль Стадник М. А.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Мазурчаку Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження механізму ізоляції пам'яті для підвищення безпеки віртуалізованих мережевих середовищах

Керівник роботи Карпінський Микола Петрович,
доктор технічних наук, професор кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » 11 2025 року № 4/7-1024

2. Термін подання студентом завершеної роботи 12.12.2025

3. Вихідні дані до роботи Архітектура віртуалізованого мережевого введення-виведення (VNIO), наявні моделі взаємодії між гіпервізором, віртуальним комутатором і віртуальними машинами, а також вимоги щодо підвищення ізоляції пам'яті та безпеки.

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз архітектури віртуалізованого мережевого введення-виведення (VNIO) та виявлення її ключових вразливостей з погляду кібербезпеки та ізоляції пам'яті.
Формування вимог до удосконаленого механізму ізоляції пам'яті у VNIO з урахуванням продуктивності та безпеки. Розроблення архітектури механізму vSwitch-to-Hypervisor (vS2H) та обґрунтування його переваг над існуючими підходами.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Актуальність дослідження. Мета, об'єкт, предмет дослідження.
Наукова новизна та практичне значення.
Завдання дослідження. Архітектура віртуалізації.
Функції, структура та принципи роботи віртуального комутатора.
Підходи до організації віртуального мережевого вводу/виводу.
Моделі обміну пам'яттю у пара-віртуалізованих VNIO. Модель безпечного обміну пам'яттю vSwitch та гіпервізор. Методика оцінювання продуктивності та експериментальне Середовище. Оцінювання продуктивності каналу даних. Оцінювання застосунків усередині віртуальних машин. Масштабованість у багатокористувацьких сценаріях. Висновки.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Охорона праці | Осухівська Г.М., к.т.н., доцент | | |
| Безпека в надзвичайних ситуаціях | Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва | | |

7. Дата видачі завдання 19.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | 20.09 – 22.09 | Виконано |
| 2. | Підбір джерел для аналізу в галузі дослідження | 25.09 – 10.10 | Виконано |
| 3. | Опрацювання джерел в галузі дослідження | 11.10 – 15.10 | Виконано |
| 4. | Налаштування симуляційного середовища | 16.10 – 25.10 | Виконано |
| 5. | Оформлення розділу «Теоретичні основи функціонування віртуалізованих мережевих платформ» | 30.10 – 05.11 | Виконано |
| 6. | Оформлення розділу «Аналіз проблем ізоляції у віртуалізованому мережевому I/O» | 06.11 – 10.11 | Виконано |
| 7. | Оформлення розділу «Експериментальна оцінка продуктивності механізму VS2H» | 11.11 – 25.11 | Виконано |
| 8. | Виконання завдання до підрозділу «Охорона праці та безпека в надзвичайних ситуаціях» | 26.11-01.12 | Виконано |
| 9. | Оформлення кваліфікаційної роботи | 02.12 – 10.12 | Виконано |
| 10. | Нормоконтроль | 08.12 – 10.12 | Виконано |
| 11. | Перевірка на плагіат | 12.12 – 14.12 | Виконано |
| 12. | Попередній захист кваліфікаційної роботи | 15.12 – 20.12 | Виконано |
| 13. | Захист кваліфікаційної роботи | 24.12.2025 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ Мазурчак Д. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінський М. П.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Дослідження механізму ізоляції пам’яті для підвищення безпеки віртуалізованих мережевих середовищах // ОР «Магістр» // Мазурчак Дмитро Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2025 // С. 82, рис. – 18, табл. – - , кресл. – 14, додат. – 1.

Ключові слова: Віртуалізація, мережеве введення-виведення, VNIO; гіпервізор, віртуальний комутатор, ізоляція трафіку, кібербезпека, продуктивність.

У кваліфікаційній роботі магістра досліджено проблему підвищення кібербезпеки у віртуалізованих мережевих платформах шляхом удосконалення механізмів ізоляції мережевого введення-виведення. Актуальність роботи визначається зростанням внутрішніх кіберзагроз та обмеженою ефективністю традиційних засобів захисту у середовищах з віртуалізацією. Проаналізовано архітектуру VNIO, виявлено ключові вразливості та сформовано вимоги до механізмів захисту на рівні гіпервізора. Запропоновано та реалізовано удосконалену архітектуру vSwitch-to-Hypervisor (vS2H), яка переносить частину функцій обробки пакетів до гіпервізора, підвищуючи ізоляцію між віртуальними машинами та стійкість до атак noisy-neighbor і внутрішніх атак. Проведено експериментальне оцінювання продуктивності, що підтвердило ефективність запропонованого підходу. Практичне значення роботи полягає у можливості впровадження механізму vS2H у хмарні й корпоративні інфраструктури.

# ABSTRACT

Research of memory isolation mechanisms to enhance security in virtualized network environments // Thesis of educational level "Master"// Dmytro Mazurchak // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБм-61 // Ternopil, 2025 // p. 82, figs. 18, tbls. -, drws. 14, apps. 1.

Keywords: virtualization, network input/output, VNIO, hypervisor, virtual switch, traffic isolation, cybersecurity, performance.

In this master’s thesis, the problem of enhancing cybersecurity in virtualized network platforms is investigated through the improvement of network input/output isolation mechanisms. The relevance of the study is determined by the growing number of internal cyber threats and the limited effectiveness of traditional security measures in virtualized environments. The VNIO architecture is analyzed, key vulnerabilities are identified, and requirements for protection mechanisms at the hypervisor level are formulated. An improved vSwitch-to-Hypervisor (vS2H) architecture is proposed and implemented, transferring part of the packet processing functionality to the hypervisor, which enhances isolation between virtual machines and increases resilience to noisy-neighbor and internal attacks. Experimental performance evaluation has confirmed the effectiveness of the proposed approach. The practical significance of the work lies in the possibility of integrating the vS2H mechanism into cloud and corporate infrastructures.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

| | | |
|---|---|---|
| VNIO | — | Virtualized Network I/O |
| SDN | — | Software-Defined Networking |
| DDoS | — | Distributed Denial of Service |
| vS2H | — | vSwitch-to-Hypervisor |
| VM | — | Virtual Machine |
| IaaS | — | Infrastructure as a Service |
| KVM | — | Kernel-based Virtual Machine |
| QEMU | — | Quick Emulator |
| TAP | — | Terminal Access Point |
| DPDK | — | Data Plane Development Kit |
| PMD | — | Polling Mode Driver |
| MAC | — | Media Access Control |
| VLAN | — | Virtual Local Area Network |
| NIC | — | Network Interface Card |
| EPT | — | Extended Page Tables |
| NPT | — | Nested Page Tables |
| DMA | — | Direct Memory Access |
| DoS | — | Denial of Service |
| QoS | — | Quality of Service |
| vCPU | — | Virtual Central Processing Unit |
| VM2vS | — | VM-to-vSwitc |
| vS2VM | — | vSwitch-to-VM |
| PD | — | Packet Delivery |

# ВСТУП

**Актуальність теми.** Стрімке зростання обсягів мережевого трафіку, широке впровадження хмарних сервісів та масова віртуалізація обчислювальних ресурсів призводять до появи нових ризиків у сфері кібербезпеки. Одним із найбільш вразливих елементів сучасних інфраструктур є підсистема віртуалізованого мережевого введення-виведення (VNIO), яка визначає взаємодію між гіпервізором, віртуальними машинами та віртуальними комутаторами. Низка відомих кіберзагроз (порушення ізоляції ресурсів, внутрішні DoS-атаки, неконтрольоване споживання процесорного часу vSwitch, побічні канали та перехоплення трафіку) породжуються саме слабкими місцями VNIO. Захист на рівні гіпервізора та віртуального комутатора стає ключовою умовою забезпечення конфіденційності, цілісності та доступності сервісів. Традиційні засоби мережевої безпеки та SDN-фаєрволи із перевіркою стану з'єднання часто не враховують внутрішні механізми обробки пакетів та розподіл навантаження між VNIO-компонентами, що створює «точку входу» для кіберзлочинців. Тому постає необхідність удосконалення архітектури VNIO з урахуванням вимог кібербезпеки з підвищення ізоляції трафіку та забезпечення передбачуваності роботи в умовах атак і пікових навантажень.

**Мета і задачі дослідження.** Метою роботи є підвищення рівня кібербезпеки віртуалізованих мережевих платформ шляхом удосконалення механізму ізоляції мережевого введення-виведення на основі архітектури vS2H та експериментальної оцінки його ефективності.

**Для досягнення мети необхідно розв'язати такі задачі:**

- проаналізувати сучасні архітектури VNIO та механізми віртуалізованого комутування з позицій ризиків кібербезпеки та ізоляції трафіку;
- визначити основні загрози та вразливості VNIO, що виникають у середовищах з SDN-керуванням та багатокористувацькими хмарними платформами;
- розробити архітектурну модель механізму vS2H, орієнтовану на посилення ізоляції, стабільності та захищеності обробки пакетів у VNIO;

- сформувати методику експериментальної оцінки продуктивності запропонованого механізму;
- реалізувати прототип механізму vS2H у тестовому середовищі та провести вимірювання продуктивності, поведінки під навантаженням;
- порівняти результати роботи механізму vS2H із класичними підходами комутації у віртуалізованих платформах і сформувати рекомендації щодо практичного впровадження.

**Об’єкт дослідження.** Віртуалізовані мережеві платформи у хмарних та корпоративних середовищах, що використовують гіпервізори, віртуальні комутатори та програмно-визначені мережі.

**Предмет дослідження**. Механізми ізоляції та обробки мережевого введення-виведення у VNIO-шляху, зокрема архітектура механізму vS2H та її вплив на продуктивність і рівень безпеки.

**Наукова новизна одержаних результатів кваліфікаційної роботи.** Удосконалено архітектурну модель механізму ізоляції мережевого введення-виведення, який переносить частину функцій обробки пакетів з віртуального комутатора безпосередньо до гіпервізора, що забезпечує вищий рівень ізоляції між віртуальними машинами.

**Практичне значення одержаних результатів.** Результати роботи мають прикладну цінність для підвищення рівня кібербезпеки та надійності хмарних інфраструктур. Розроблений механізм vS2H може бути інтегрований у гіпервізори та vSwitch-платформи для зменшення ризику порушення ізоляції між орендарями. Напрацювання можуть бути впроваджені в освітній процес у дисциплінах з віртуалізації, мережевої безпеки, хмарних технологій.

**Апробація результатів магістерської роботи.** Основні результати дослідження були представлені на XIII науково-технічній конференції «Інформаційні моделі, системи та технології» (ТНТУ, Тернопіль, Україна, 17-18 грудня 2025 р).

**Публікації.** Основні результати кваліфікаційної роботи опубліковано у працях конференції (див. Додаток А).

# РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ФУНКЦІОНУВАННЯ ВІРТУАЛІЗОВАНИХ МЕРЕЖЕВИХ ПЛАТФОРМ

## 1.1 Сутність і еволюція технологій віртуалізації

Віртуалізація є фундаментальною технологією, на якій базуються сучасні хмарні обчислення та багатокористувацькі інфраструктури центрів обробки даних [1]. Її сутність полягає в абстрагуванні фізичних апаратних ресурсів серверів (процесорів, оперативної пам’яті, підсистем зберігання даних та мережевих інтерфейсів) у вигляді логічних, ізольованих середовищ виконання, що сприймаються операційними системами як “фізичні” машини. Такі логічні сутності прийнято називати віртуальними машинами (VM). Кожна VM має власну гостьову операційну систему, програмне забезпечення і стек протоколів, тоді як доступ до реальних ресурсів здійснюється опосередковано через шар віртуалізації. На рисунку 1.1 показано типовий приклад хмарної платформи, де на комерційному сервері з певною кількістю ядер CPU, обсягом оперативної пам’яті та дисковим сховищем одночасно функціонує декілька VM з різними операційними системами та прикладними навантаженнями.

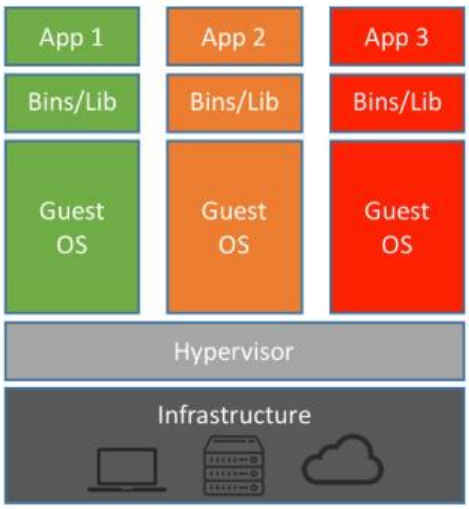



Рисунок 1.1 – Архітектура віртуалізації

Саме завдяки шару віртуалізації між апаратною платформою та гостьовими ОС ці VM отримують ізольовані віртуальні ресурси, хоча фізично поділяють одні й ті самі компоненти обладнання [2].

З історичної точки зору, ідеї віртуалізації виникли задовго до появи хмарних обчислень. Перші реалізації часу поділу та віртуальних машин з'явилися ще в епоху мейнфреймів, коли дорогі обчислювальні системи потрібно було розділити між багатьма користувачами. Пізніше, із здешевленням апаратури та поширенням одноцільових серверів, інтерес до віртуалізації тимчасово зменшився. Новий етап розвитку почався тоді, коли зріс попит на консолідацію серверів і зменшення витрат на експлуатацію дата-центрів, а також на гнучке використання ресурсів. Саме в цей період було запропоновано сучасні платформи віртуалізації для x86-архітектури, які розв'язали низку технічних обмежень процесорів загального призначення і відкрили шлях до масового впровадження віртуальних машин у промислових системах.

У контексті хмарних платформ, які, починаючи приблизно з 2005 року, активно розвиваються як економічно ефективна та гнучка модель обчислень, віртуалізація перетворилася на базовий механізм реалізації моделі інфраструктура як сервіс (IaaS) (див. рисунок 1.2) [3].

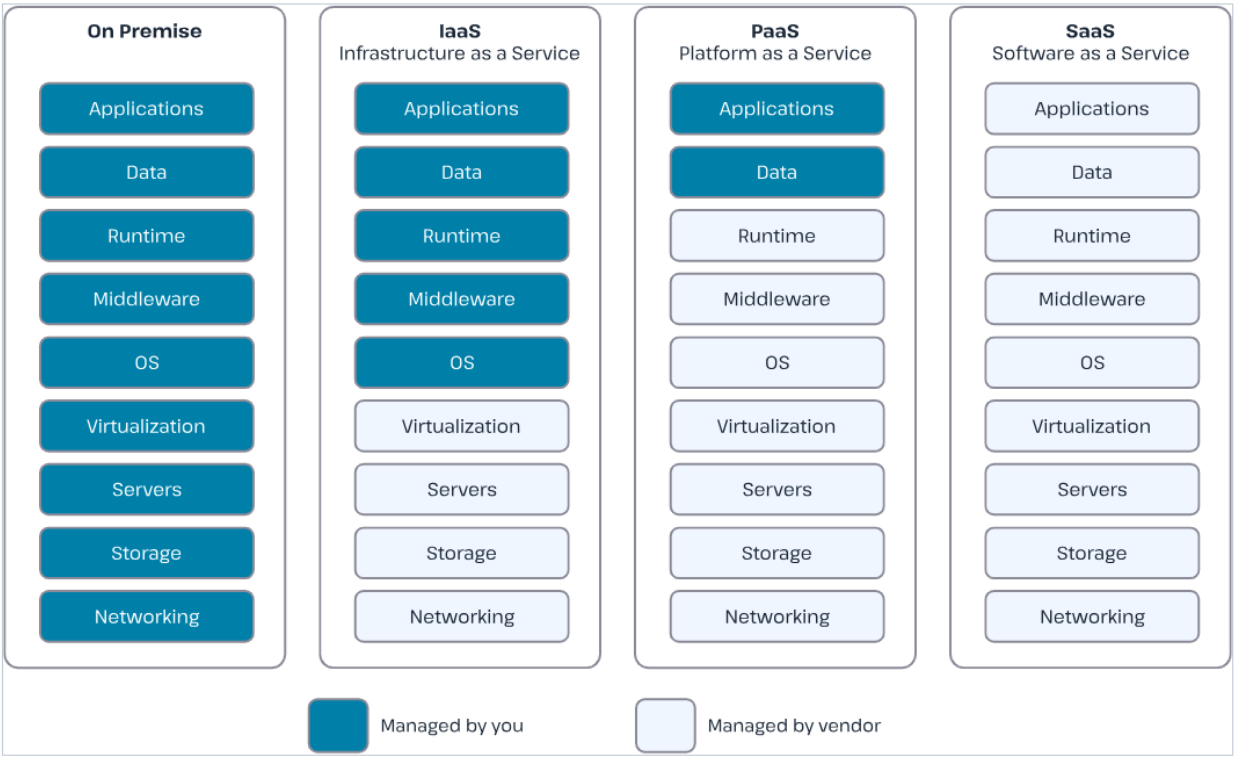



Рисунок 1.2 – Моделі надання сервісів в хмарних середовищах

Завдяки їй провайдери хмарних сервісів можуть розміщувати велику кількість незалежних орендованих VM на одному фізичному сервері, надаючи кожному орендарю ізольоване середовище виконання із власними гарантіями продуктивності та доступності. На концептуальному рівні це добре ілюструє рисунок 1.1, де між апаратними ресурсами та множиною VM знаходиться проміжний шар - гіпервізор, який здійснює відображення фізичних ресурсів у віртуальні та контролює їх розподіл.

Ключовою властивістю віртуалізації є ізоляція. З погляду користувача, VM сприймається як окремий логічний сервер, що має власні процесори, пам’ять, диски та мережеві інтерфейси, тоді як фактично вона є лише процесом на стороні хоста. Гіпервізор перехоплює привілейовані інструкції гостьової ОС, емулює або відображає апаратні ресурси та забезпечує, щоб випадкові або зловмисні дії однієї VM не впливали на інші. Унаслідок цього досягається ефект одночасної підтримки багатьох ізольованих середовищ на спільній фізичній платформі.

Архітектура сучасних платформи віртуалізації реалізуються як поєднання компонентів у просторі ядра та в просторі користувача. На рисунку 1.3 показано два поширені приклади  QEMU/KVM та Xen [4].

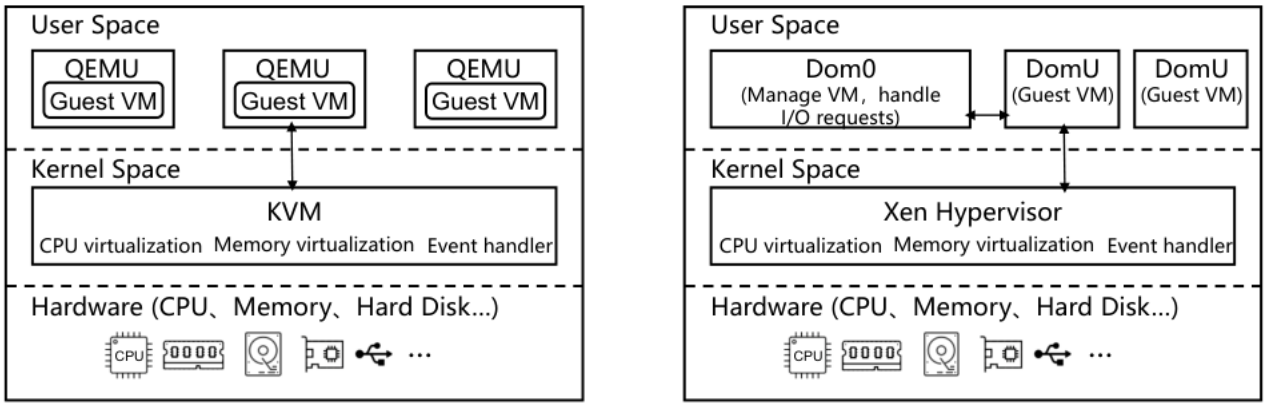



Рисунок 1.3 – Гіпервізор KVM та XEN

У випадку QEMU/KVM частина гіпервізора працює в ядрі, забезпечуючи підтримку апаратної віртуалізації процесора та пам’яті, тоді як у просторі користувача функціонують окремі процеси QEMU, що емулюють пристрої та обробляють ввід/вивід для кожної VM [5-7]. У випадку Xen виділяється

спеціалізований гіпервізор у ядрі та окремий привілейований домен Dom0 для обробки запитів вводу/виводу й управління іншими гостьовими доменами [8]. Саме поєднання компонентів у ядрі та користувацькому просторі забезпечує тонкий контроль над апаратними ресурсами й можливість прозорої їх віртуалізації.

Еволюція віртуалізації на рівні апаратури істотно вплинула на продуктивність і прозорість роботи VM. На ранніх етапах платформи були змушені використовувати техніки бінарної трансляції або пара-віртуалізації, коли гостьова ОС модифікувалася для взаємодії з гіпервізором. З появою апаратної підтримки віртуалізації, такої як розширення Intel VT-x та AMD-V [9], гіпервізори отримали можливість виконувати гостьові операційні системи практично без модифікації, передаючи більшість інструкцій напряму на процесор, а перехоплюючи лише привілейовані операції. Це дало змогу суттєво знизити накладні витрати й наблизити продуктивність VM до рівня ‘bare-metal’, що особливо важливо для хмарних сценаріїв з великою кількістю високонавантажених сервісів.

З погляду функцій, які реалізує шар віртуалізації, можна виділити декілька ключових напрями. По-перше, це віртуалізація процесорних ресурсів, де фізичні ядра CPU подаються гостьовим системам у вигляді віртуальних процесорів, між якими гіпервізор здійснює планування. По-друге, це віртуалізація пам’яті, що включає підтримку віртуальних адресних просторів VM, організацію таблиць сторінок та механізми двоступеневої трансляції адрес, які забезпечують прозоре відображення гостьової пам’яті на фізичні сторінки. По-третє, це віртуалізація вводу/виводу, де реальні пристрої (мережеві адаптери, диски, контролери) презентуються гостьовим ОС як віртуальні, а операції з ними проходять через гіпервізор чи спеціалізовані драйвери.

Подальший розвиток віртуалізації був зумовлений не лише потребою в кращій продуктивності, а й вимогами до гнучкого управління ресурсами та масштабуванням у великих хмарних інфраструктурах. Поступово концепція “віртуальної машини” почала доповнюватися іншими формами логічної ізоляції, зокрема контейнерною віртуалізацією на рівні операційної системи. Проте для

задач, де важливою є жорстка ізоляція орендарів, розмежування привілеїв та підтримка різнорідних гостьових ОС, саме гіпервізорна віртуалізація залишається базовою технологією. У такій моделі провайдер може гарантувати, що з погляду безпеки та керування ресурсами кожна VM розглядається як окремий об’єкт із чітко визначеними межами доступу до ресурсів процесора, пам’яті, диску та мереже.

Разом із цим, збільшення щільності розміщення віртуальних машин на одному фізичному сервері висунуло нові вимоги до ефективності реалізації віртуалізації. Гіпервізор повинен не лише забезпечувати коректність та ізоляцію, а й мінімізувати накладні витрати на перемикання контексту, обробку переривань та маршрутизацію операцій вводу/виводу. У хмарних платформах, орієнтованих на сотні й тисячі VM, саме здатність масштабуватися без суттєвих втрат продуктивності визначає практичну цінність технології віртуалізації. Тому еволюція гіпервізорів відбувалася у напрямі оптимізації критичних шляхів доступу до ресурсів, удосконалення алгоритмів планування віртуальних процесорів, удосконалення механізмів керування пам’яттю та інтеграції з підсистемами зберігання й мережі.

У контексті мережевої взаємодії віртуалізація створює ще один важливий вимір - потребу в логічному відокремленні мережевих ресурсів орендарів безпосередньо на хост-сервері. Хоча детальна архітектура мережевої віртуалізації та програмно реалізованого віртуального комутатора (vSwitch [10]) буде розглянута в наступних підрозділах, на рівні сутності технологій важливо підкреслити, що мережеві можливості VM теж є результатом віртуалізації. Для гостьової системи мережевий інтерфейс виглядає як повноцінна мережева карта, проте всі операції з передачі і приймання пакетів фактично реалізуються програмними компонентами хоста, які забезпечують мультиплексування спільного фізичного адаптера між багатьма орендарями.

Таким чином, еволюція технологій віртуалізації пройшла шлях від базового розподілу обчислювального часу на мейнфреймах до комплексних платформ, здатних одночасно віртуалізувати процесор, пам’ять, підсистеми зберігання та мережу для великої кількості незалежних користувачів. Гіпервізор став основою

хмарних обчислень, дозволивши провайдерам ефективно консолідувати навантаження та гнучко керувати ресурсами дата-центрів.

### 1.2 Функції, структура та принципи роботи віртуального комутатора

Віртуальний комутатор, або vSwitch, є ключовим компонентом програмно визначеної мережевої інфраструктури хмарних обчислень [10]. Він забезпечує логічне об’єднання віртуальних машин у віртуальні мережі, здійснює програмну обробку трафіку та керує маршрутами передачі пакетів між VM і зовнішньою фізичною мережею. На відміну від традиційних апаратних комутаторів, які працюють безпосередньо з фізичними інтерфейсами й реалізовані спеціалізованими мікросхемами, vSwitch є програмним компонентом, розміщеним у просторі користувача й повністю керованим операційною системою хоста або віртуалізаційною платформою. Саме він виконує функції мережевої віртуалізації, дозволяючи розділяти спільний фізичний мережевий інтерфейс між багатьма незалежними віртуальними машинами.

Перші концепти віртуального комутатора походять від механізму Linux Bridge (див. рисунок 1.4) [11].

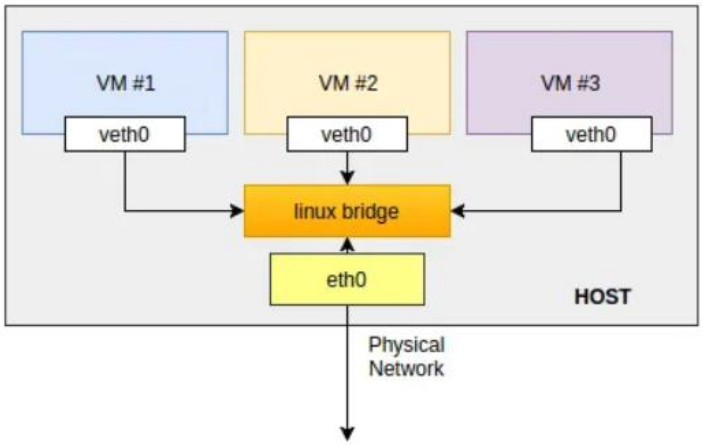



Рисунок 1.4 – Linux Bridge

Linux Bridge являє собою програмну реалізацію комутатора другого рівня, що функціонує в ядрі операційної системи. Він може пов’язувати між собою різні мережеві інтерфейси, включно з фізичними NIC та віртуальними TAP-пристроями, створюючи логічну ланку, через яку віртуальні машини

взаємодіють із мережею. Попри свою простоту, Linux Bridge має два фундаментальні обмеження. По-перше, передача пакетів між TAP-пристроями й фізичною мережею потребує багаторазового копіювання даних у просторі ядра, що суттєво знижує продуктивність. По-друге, Bridge не має гнучких механізмів моніторингу та конфігурації, які потрібні для реалізації складних мережевих політик у хмарних середовищах. Ці недоліки зумовили появу нової генерації програмних комутаторів - високопродуктивних і гнучких vSwitch-систем.

Еволюція від Linux Bridge до повноцінного vSwitch відображена на рисунку 1.5.

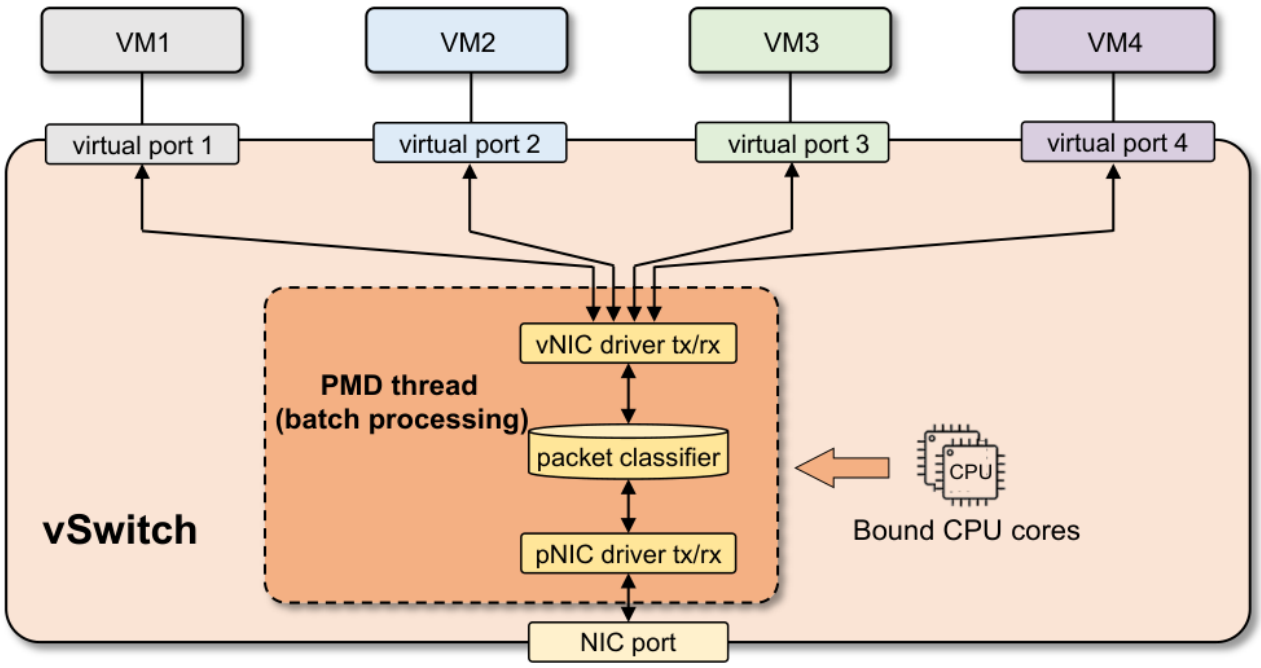



Рисунок 1.5 – vSwitch

Основний принцип полягає у перенесенні обробки пакетів із простору ядра до простору користувача. Такий підхід дозволяє уникнути зайвих копіювань, використовувати високооптимізовані бібліотеки прискореної обробки пакетів, як-от DPDK, та реалізувати гнучкі механізми програмного керування мережевою інфраструктурою. У vSwitch усі етапи обробки трафіку (від взаємодії з драйвером NIC до маршрутизації та застосування політик) виконуються в межах одного користувацького процесу. Завдяки цьому структура vSwitch змогла інтегрувати функції, які раніше були розподілені між ядром ОС, драйверами та зовнішніми мережевими пристроями.

Структура сучасного віртуального комутатора включає порти, віртуальні драйвери, модуль класифікації пакетів, таблиці потоків і модулі передачі трафіку. На рисунку 1.5 показано типовий вигляд vSwitch, який використовується у провідних хмарних середовищах. Кожен віртуальний або фізичний інтерфейс, що належить VM або NIC, під'єднано до vSwitch у вигляді окремого порту. Призначений для обробки трафіку модуль PMD працює в режимі активного опитування портів, обходячи їх циклічно й обробляючи пакети партіями, що суттєво знижує накладні витрати на перемикання контексту. Такий підхід дозволяє забезпечити високу пропускну здатність і низькі затримки порівняно з традиційними перериваннями NIC-драйвера.

Однією з ключових функцій vSwitch є віртуалізація мережевого вводу/виводу. Коли віртуальна машина передає пакет, він надходить до vSwitch через віртуальний мережевий інтерфейс, зазвичай це TAP або vhost-user порт, залежно від технологічної реалізації. У межах vSwitch пакет копіюється у внутрішній буфер, після чого аналізується модулем класифікації. Цей модуль здійснює розбір заголовків і визначає відповідний запис у таблиці потоків (flow table), який містить набір правил типу 'match–action' - підхід, успадкований від SDN-архітектури. У разі успішного збігу vSwitch виконує відповідні дії: перенаправлення на інший порт, застосування обмежень пропускної здатності, модифікацію заголовків або інші операції.

Відмінність віртуального комутатора від Linux Bridge полягає не лише в просторовому розташуванні обробки, а й у можливості працювати з гнучкими та ієрархічними структурами таблиць потоків. У SDN-парадигмі vSwitch реалізує базовий принцип, що дозволяє хмарним провайдерам контролювати параметри мережевої підсистеми на рівні окремих портів, потоків і пакетів. У той час як Linux Bridge дозволяв лише найпростіші дії з MAC-орієнтованого комутування, сучасні vSwitch-системи підтримують фільтрацію за IP-префіксами, VLAN - ідентифікаторами, транспортними заголовками та іншими характеристиками. Це забезпечує гнучке програмне керування мережею, що є важливою складовою масштабованих хмарних інфраструктур.

Завдяки переносу обробки пакетів у простір користувача vSwitch отримує можливість використовувати високопродуктивні бібліотеки, як-от DPDK [12]. Це дає змогу працювати без втручання ядра, скорочуючи кількість копіювань і зменшуючи затримки. DPDK забезпечує прямий доступ до апаратних NIC через механізми kernel bypass, що дозволяє досягати пропускних здатностей, близьких до лінійних швидкостей фізичних адаптерів. PMD-потоки працюють із портами vSwitch без участі ядра ОС, зчитуючи та передаючи пакети безпосередньо у програмний пайплайн обробки.

Принципи роботи vSwitch формуються як поєднання пакетно-орієнтованої логіки та потокової семантики. На вхід vSwitch надходять пакети, які об’єднуються у (групи пакетів) батчі, що дозволяє оптимізувати доступ до кеш-пам’яті та прискорити пошук у таблицях потоків. Модуль класифікації використовує структуровані багаторівневі таблиці для пошуку відповідного правила. Після класифікації пакет отримує набір дій і передається до модуля передачі, який надсилає його на відповідний порт. Усі ці операції виконуються у високопродуктивному циклі PMD-потоку, що працює без переривань, скануючи порти та обробляючи черги пакетів.

Функціональна роль vSwitch виходить далеко за межі простого переспрямування пакетів. vSwitch працює як повноцінний програмно визначений елемент, який взаємодіє з контролерами SDN через протоколи на кшталт OpenFlow [13]. На відміну від апаратних комутаторів, які мають фіксовані таблиці й обмеження апаратних ресурсів, програмний vSwitch може динамічно змінювати таблиці правил, збільшувати чи зменшувати їх кількість та адаптуватися до потреб кожного клієнта.

Однак гнучкість та програмна природа vSwitch одночасно створюють нові виклики. Оскільки всі VM на фізичному сервері використовують спільний vSwitch для обробки трафіку, продуктивність і затримки можуть змінюватися залежно від навантаження. Більше того, vSwitch є високопривілейованим процесом, який має доступ і до пам’яті гостьових VM (через механізми VNIO), і до драйверів NIC, і до спільних структур даних. Це означає, що будь-яке перевантаження або помилка в модулі класифікації чи в таблиці потоків може

вплинути на всі VM, що обробляються тим самим vSwitch. Важливо зазначити, що vSwitch одночасно виконує обов’язки комутатора, маршрутизатора, монітора та механізму політик.

У підсумку vSwitch є критичним елементом архітектури хмарних платформ, який поєднує програмну гнучкість із високою продуктивністю, забезпечує ефективну віртуалізацію мережевих ресурсів, дозволяє ізолювати логічні мережі орендарів і реалізувати складні політики керування трафіком.

### 1.3 Віртуалізація ресурсів CPU, пам’яті, мережі та сховищ

Віртуалізація ресурсів у хмарних обчисленнях є фундаментальним механізмом, який забезпечує можливість одночасної роботи великої кількості ізольованих віртуальних машин на спільній фізичній інфраструктурі. Один комерційний сервер із певною кількістю фізичних ядер процесора, обсягом оперативної пам’яті та дисковим сховищем забезпечує роботу багатьох VM, кожна з яких сприймає виділені їй ресурси як власні. Цей ефект досягається завдяки гіпервізору, який виступає посередником між апаратною платформою та гостьовими операційними системами. Віртуалізація ресурсів охоплює чотири основні категорії: CPU, пам’ять, мережеву підсистему та підсистему зберігання. Кожна з них має власні механізми абстрагування, планування та контролю доступу, які разом формують єдину модель ізоляції та ефективного використання обладнання.

Віртуалізація процесорних ресурсів є першою й найважливішою складовою роботи гіпервізора. У межах VM гостьова ОС бачить набір віртуальних процесорів, аналогічних фізичним, але фактично виконання цих віртуальних ядер здійснюється шляхом планування на фізичних ядрах CPU. Гіпервізор перехоплює привілейовані інструкції, забезпечує коректність роботи таймерів, реалізує віртуальні переривання та підтримує ізольоване виконання кожної VM у власному адресному просторі. У варіанті QEMU/KVM модуль KVM у ядрі оперує безпосередньо апаратними інструкціями віртуалізації, тоді як процеси QEMU виконують роль контейнерів для гостьових ОС. В архітектурі Xen

гіпервізор повністю контролює планування віртуальних процесорів, а Dom0 обробляє частину операцій вводу/виводу. Незалежно від моделі, CPU-віртуалізація має забезпечити два ключові ефекти: по-перше, ізоляцію, що виключає вплив однієї VM на іншу; по-друге, гнучке планування, яке дозволяє провайдеру розподіляти обчислювальні ресурси максимально ефективно. Важливою частиною CPU-віртуалізації є апаратна підтримка, така як Intel VT-x чи AMD-V, які реалізують механізми переведення гостьової ОС у невіртуалізований режим із можливістю перехоплення критичних операцій. Такі розширення значно зменшують накладні витрати, пов’язані з виконанням гостьового коду, і забезпечують продуктивність, наближену до фізичної. Гіпервізор періодично перемикає контекст між віртуальними ядрами, що дозволяє десяткам і навіть сотням VM розділяти обчислювальні ресурси одного сервера. На відміну від традиційної багатозадачності в операційній системі, гіпервізор має контролювати привілейовані операції VM та гарантувати, що жодна гостьова ОС не зможе виконати інструкцію, яка порушує коректність роботи інших. Цей рівень контролю є основою безпеки й стабільності хмарної інфраструктури.

Віртуалізація пам’яті доповнює CPU-віртуалізацію, забезпечуючи для кожної VM власний віртуальний адресний простір. Гостьова ОС працює з уявленням, що має повноцінний контроль над усією виділеною їй пам’яттю, хоча фактично фізичні сторінки розподіляються та відображаються гіпервізором. У цьому механізмі ключову роль відіграє двоступенева трансляція адрес, коли спочатку відбувається перехід від віртуальної адреси гостьової ОС до гостьової фізичної адреси, а потім її відображення в реальну фізичну адресу на стороні хоста. Апаратні можливості, такі як EPT або NPT, суттєво пришвидшують цей процес, зменшуючи кількість пасток (traps) у гіпервізор. Оскільки пам’ять є одним з найбільш критичних ресурсів, гіпервізор забезпечує захист сторінок кожної VM, не допускаючи доступу за межі виділеного обсягу.

У навантажених середовищах віртуалізація пам’яті вимагає механізмів оптимізації, таких як ballooning та deduplication. Ballooning дає змогу динамічно регулювати обсяг пам’яті, доступний VM, залежно від потреб та політик

хмарного провайдера. Deduplication дозволяє об’єднувати однакові фізичні сторінки, що виникають, наприклад, при запуску великої кількості однотипних VM.

Віртуалізація дискової підсистеми забезпечує гостьовим ОС уявлення про власний блоковий пристрій. У QEMU/KVM обробка операцій зберігання даних здійснюється процесом QEMU у просторі користувача. У Xen цю роль виконує привілейований домен Dom0. У сучасних системах використовуються різні типи віртуальних дисків: образи файлів (qcow2, raw), логічні томи чи відображення на фізичні SSD/NVMe. Оскільки операції з диском мають значно вищу затримку, ніж операції CPU чи пам’яті, важливу роль відіграють кеші, асинхронні черги та драйвери на кшталт virtio-blk або virtio-scsi. Вони дозволяють зменшити накладні витрати на ввід/вивід, працювати з декількома чергами та балансувати навантаження між ними.

Віртуалізація мережевих ресурсів є найскладнішим аспектом, оскільки потребує не лише абстракції фізичного мережевого інтерфейсу, а й реалізації логічно ізольованих віртуальних мереж між VM. На вищому рівні гостьова ОС бачить власний мережевий інтерфейс, який може бути реалізований через TAP-пристрої, virtio-net або vhost-user, залежно від платформи. Фактична маршрутизація пакетів здійснюється віртуальним комутатором (vSwitch). Він обробляє пакети в просторі користувача, здійснює класифікацію за правилами типу match–action та керує передаванням трафіку між VM і фізичною мережею. На цьому рівні мережеві ресурси стають повністю програмно визначеними, що дає можливість застосовувати політики безпеки, балансування навантаження та моніторинг на рівні потоків.

Важливим моментом є інтеграція мережевої віртуалізації з апаратними прискорювачами. Kernel bypass-технології, такі як DPDK, дозволяють vSwitch працювати без участі ядра ОС, зменшуючи накладні витрати та затримки. Це критично важливо, оскільки мережеве навантаження в сучасних дата-центрах може досягати сотень гігабіт на секунду, і програмні компоненти повинні забезпечувати прийнятний рівень масштабування. PMD-потоки виконують

активне опитування портів, працюючи в режимі high-throughput, що дозволяє обробляти великі обсяги трафіку без втрати пакетів.

Віртуалізація сховищ у хмарній платформі охоплює не лише логічні диски VM, а й мережеві файлові системи, блочні сховища, реплікацію та розподілені файлові системи. Гостьова ОС, взаємодіючи з віртуальними пристроями, працює з ними так, ніби це звичайні локальні диски, хоча насправді вони можуть бути розташовані у віддаленій мережевій підсистемі зберігання. Гіпервізор відповідає за коректне перенаправлення запитів вводу/виводу та забезпечення ізоляції між VM. У масштабованих інфраструктурах доцільно використовувати прискорювачі на кшталт NVMe-over-TCP або RDMA, але незалежно від технології віртуальний пристрій завжди має бути повністю ізольованим від інших.

У підсумку віртуалізація CPU, пам’яті, мережі та сховищ формує комплексний механізм абстрагування фізичної інфраструктури, який дозволяє одночасно розміщувати численні ізольовані робочі навантаження.

## 1.4 Проблема ізоляції ресурсів і вплив спільного використання CPU/пам’яті на QoS та безпеку

Проблема ізоляції ресурсів у віртуалізованих середовищах має подвійний вимір та впливає на продуктивність та безпеку. Якщо на рівні продуктивності конкуренція за спільні ресурси призводить до коливань затримок, зниження пропускної здатності або порушення SLA, то на рівні безпеки ті самі механізми спільного доступу створюють небезпечні побічні канали, які можуть бути використані для атак на конфіденційність і цілісність роботи системи. Сучасні хмарні платформи, де на одному фізичному сервері працює велика кількість віртуальних машин, мають мінімізувати ризик витоку даних або несанкціонованого впливу одного орендаря на іншого. Проте забезпечити повну ізоляцію апаратних ресурсів у межах програмної віртуалізації напряму неможливо, що формує складний комплекс викликів.

На рівні CPU гіпервізор створює логічно ізольовані віртуальні процесори, проте фізичні ядра та їхні кеші залишаються спільними. Декілька VM формально

мають власні CPU-ресурси, але фактично вони виконуються на тих самих ядрах і використовують одну й ту саму ієрархію кеш-пам’яті. Цей шар кешів L1/L2/L3 давно відомий як потенційний об’єкт атак типу side-channel, де одна VM вимірює затримки доступу до кешу, щоб отримати інформацію про активність іншої VM. Атаки класу Prime+Probe, Flush+Reload, Spectre, Meltdown і L1TF неодноразово демонстрували, що навіть у моделях жорсткої гіпервізорної ізоляції можливо виявити статистичні характеристики обчислень, що виконуються в інших віртуальних середовищах [14]. Це означає, що конкуренція за спільні CPU-ресурси не лише порушує QoS, а й створює фундаментальний ризик компрометації даних.

Проблема посилюється тоді, коли розглянути взаємодію віртуальних машин із віртуальним комутатором (vSwitch), що працює у просторі користувача й використовує виділені ядра CPU. PMD-потоки vSwitch опитують порти, виконують класифікацію пакетів та їх пересилання. Усі ці операції відбуваються в одному високопривілейованому процесі. Якщо одна VM генерує специфічний трафік із високою інтенсивністю, вона може цілеспрямовано навантажити ядро, на якому працює vSwitch, створюючи побічні затримки для інших VM. Обробка пакетів стає передбачуваним джерелом side-channel взаємодії [15]. Злоумисник може вимірювати час обробки власних пакетів, щоб визначити активність інших віртуальних машин, їхню інтенсивність трафіку або навіть характер мережевої поведінки.

Ще складнішою є взаємодія між VM та vSwitch на рівні пам’яті. У сучасних моделях прискореного VNIO використовується спільна пам’ять, яка забезпечує копіювання пакетів між пам’яттю VM та буфером vSwitch. У платформах QEMU/KVM і Xen ключові компоненти доступу до пам’яті знаходяться або у процесах QEMU, або в привілейованому домені Dom0. Саме ці компоненти взаємодіють із механізмами VNIO, що створює небезпечне середовище. Спільні буфери, кільцеві черги, DMA-вікна та віртуальні драйвери стають точками доступу, де помилка валідації або вразливість у vSwitch можуть адекватно експлуатуватися для атак на пам’ять інших VM. Дослідження RedHat підтвердили існування прямих DMA-атак, коли користувач із правами доступу

до однієї VM отримував можливість нелегального читання та модифікації пам’яті іншої VM через спільні структури VNIO [16]. Це означає, що навіть у традиційних моделях гіпервізорної ізоляції спільні механізми взаємодії між VM і vSwitch створюють значні ризики.

Крім того, спільне використання пам’яті не обмежується VNIO. Механізми оптимізації, такі як deduplication або ballooning, також порушують природну ізоляцію сторінок пам’яті. Deduplication, що об’єднує однакові сторінки між VM для зменшення витрат пам’яті, був визнаний вразливим до атак типу Copy-on-write side-channel, у яких зловмисник може робити висновки про вміст пам’яті інших VM. Через це багато хмарних провайдерів відмовились від автоматичного deduplication у багатокористувацьких середовищах, визнавши його небезпечним у контексті безпеки.

На рівні CPU конкуренція за шини пам’яті та кеш призводить не лише до падіння продуктивності, а й до можливості атак timing-based inference. Коли декілька VM конкурують за спільні ресурси, вразлива VM може фіксувати затримки доступу до пам’яті й відстежувати активність інших VM. Цей тип атак у хмарній інфраструктурі набуває особливої небезпечності, оскільки зловмисник може задіяти дешеву VM для стеження за поведінкою підприємства, яке розмістило інші VM на тому самому сервері. У багатоорендних середовищах це створює критичний рівень ризику для конфіденційності.

Аналогічна проблема виникає у контексті мережевої підсистеми. vSwitch працює на спільних CPU-ядрах, виконуючи класифікацію пакетів для всіх VM. Якщо одна VM сформує трафік, що спричиняє часті промахи в таблиці потоків (flow table), вона здатна навмисно уповільнити роботу PMD-полос, що вплине на інші VM. Це не лише деградація QoS, а й спосіб організації атак на відмову в обслуговуванні (DoS) через спільну інфраструктуру [17,18]. Низькошвидкісний трафік може заповнювати спільну таблицю потоків, підвищувати складність класифікації (tuple space explosion) та фактично знижувати продуктивність мережевої підсистеми до 20%. У цьому випадку спільність CPU та пам’яті у vSwitch створює критичну уразливість, яку складно усунути без фундаментальних змін в архітектурі.

Ізоляція ресурсів у сучасних хмарних середовищах стикається ще з однією проблемою - неможливістю точно моделювати, як навантаження однієї VM вплине на інші. Продуктивність CPU у програмних мережевих компонентах не є сталою. Вона залежить від розміру пакетів, розподілу трафіку, кількості активних портів, структури таблиць потоків і навіть від поведінки кешу. Через це традиційні механізми QoS, засновані на пропускній здатності, не можуть гарантувати безпеку, оскільки не здатні запобігти побічному впливу. У деяких випадках одна VM, яка навмисно генерує погано структурований трафік, може не лише погіршити QoS сусідів, а й створити side-channel, що дозволить пройти у фазу активної атаки.

Саме ці проблеми стають передумовою до необхідності розробки нових, більш строгих механізмів ізоляції, які враховують не лише логічний поділ, а й загрози, що випливають зі спільного використання апаратних компонентів у хмарних віртуалізованих середовищах.

### 1.5 Висновки до розділу 1

В першому розділі було розглянуто теоретичні засади функціонування віртуалізованих мережевих платформ, що формують основу подальших досліджень у роботі. Насамперед було уточнено сутність віртуалізації як механізму абстрагування фізичних ресурсів серверів у вигляді ізольованих логічних середовищ виконання, які представлено у формі віртуальних машин. Показано, що саме завдяки віртуалізації на основі гіпервізора стала можливою модель хмарних обчислень з підтримкою великої кількості орендарів, де кожна VM сприймає виділені їй обчислювальні ресурси як власні, хоча фактично вони спільно використовуються на рівні фізичної платформи. Було проаналізовано архітектурні підходи до побудови віртуалізованих систем на основі гіпервізора, зокрема на прикладі QEMU/KVM та Xen, і підкреслено роль апаратної підтримки віртуалізації (Intel VT-x, AMD-V, механізми двоступеневої трансляції адрес) у зменшенні накладних витрат і наближенні продуктивності віртуальних машин до рівня фізичних серверів. Відзначено, що гіпервізор поєднує компоненти у

просторі ядра та користувача, здійснюючи тонкий контроль над CPU, пам’яттю, підсистемами вводу/виводу та забезпечуючи ізоляцію між орендарями.

Окрему увагу було приділено віртуальному комутатору як ключовому елементу мережевої віртуалізації у хмарних середовищах. Показано еволюцію від простого Linux Bridge до високопродуктивних користувацьких vSwitch-систем, які реалізують програмно визначену логіку обробки трафіку, підтримують гнучкі правила, інтегруються з SDN-контролерами та використовують механізми kernel bypass (DPDK) для досягнення високої пропускної здатності. Було наголошено, що vSwitch виконує не лише функції комутації, а й роль централізованого елемента політик, моніторингу та управління трафіком у віртуалізованій мережі. Узагальнено підходи до віртуалізації основних типів ресурсів: процесорних, пам’яті, мережі та сховищ. Показано, що для кожного з цих механізмів застосовуються власні механізми абстрагування та планування, проте всі вони опираються на спільну ідеологію ізоляції гостьових середовищ та ефективної консолідації навантажень на фізичній інфраструктурі. Водночас у розділі було показано, що спільне використання фізичних ресурсів неминуче породжує проблеми ізоляції, які проявляються як у площині продуктивності, так і в площині безпеки. Конкуренція за ядра CPU, багаторівневі кеші та шини пам’яті призводить до нестабільності показників QoS, коливань затримок і пропускної здатності для мережевих сервісів, що особливо критично для орендованих VM із жорсткими SLA. Спільні моделі доступу до пам’яті, які застосовуються у механізмах прискореного мережевого вводу/виводу, створюють додаткові вектори атак, зокрема побічні канали та ризики нелегального доступу до пам’яті інших віртуальних машин через вразливості у vSwitch чи VNIO.

Таким чином, у першому розділі сформовано цілісне бачення архітектури віртуалізованих мережевих платформ: від базових принципів віртуалізації та ролі гіпервізора до функціонування віртуального комутатора і механізмів віртуалізації основних ресурсів. Виявлені обмеження та загрози, пов’язані зі спільним використанням CPU і пам’яті, а також з централізованою роллю vSwitch у багатокористувацькому середовищі, обґрунтовують необхідність

розроблення спеціалізованих механізмів ізоляції продуктивності, відмовостійкості та безпеки, які й розглядаються у наступних розділах роботи.

# РОЗДІЛ 2 АНАЛІЗ ПРОБЛЕМ ІЗОЛЯЦІЇ У ВІРТУАЛІЗОВАНОМУ МЕРЕЖЕВОМУ I/O

## 2.1 Архітектурні основи віртуалізованих мережевих підсистем

Архітектура віртуалізованих мережевих підсистем формує фундамент сучасних хмарних середовищ, де одночасно функціонують сотні та тисячі ізольованих віртуальних машин, що конкурують за доступ до обмежених фізичних ресурсів мережевої інфраструктури. Одним із ключових елементів цієї архітектури є механізми VNIO [19], які відповідають за транспортування пакетів між фізичним мережевим інтерфейсом хоста та віртуальними мережевими інтерфейсами гостьових операційних систем. В умовах зростаючих вимог до пропускної здатності, мінімізації затримок та дотримання суворої ізоляції між орендарями архітектура VNIO стала критично важливим аспектом загальної безпеки та продуктивності хмарних платформ. Функціонування VNIO засноване на взаємодії трьох базових елементів: гостьових операційних систем, які працюють усередині віртуальних машин; гіпервізора, що керує їхнім життєвим циклом та пам’яттю; та програмного комутатора (vSwitch), який виконує класифікацію, переспрямування та контроль трафіку між фізичними та віртуальними портами. На рисунку 2.1 показано три фундаментальні підходи до організації віртуального мережевого вводу/виводу.

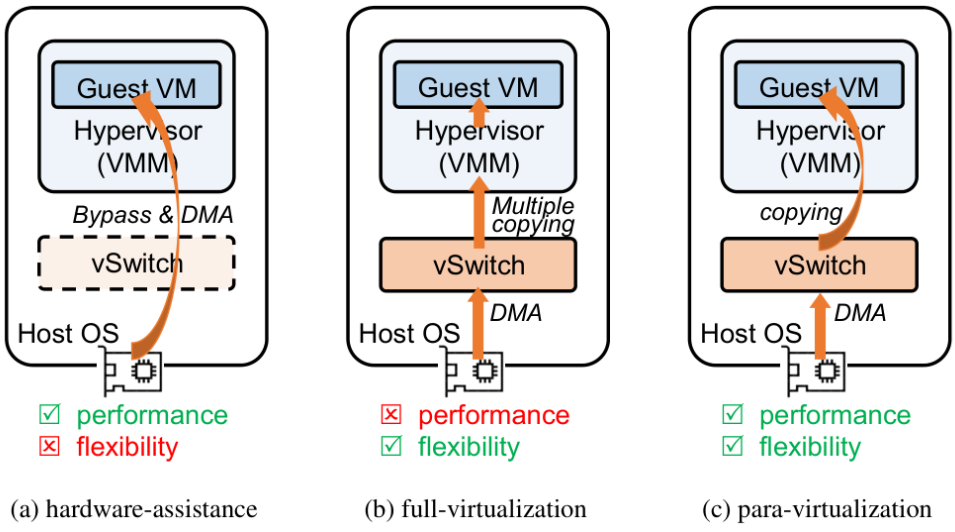



Рисунок 2.1 – Підходи до організації віртуального мережевого вводу/виводу

Що включають: апаратний, повністю віртуалізований та пара-віртуалізований. Ключова відмінність між ними полягає у ступені втручання гіпервізора у процес обробки пакетів та кількості копіювань пам’яті, які необхідні для транспортування одного пакета від NIC до VM.

У повністю віртуалізованих архітектурах гіпервізор виконує емітацію мережевого пристрою, тобто створює програмну модель фізичного NIC, яку драйвер гостьової ОС сприймає як звичайний апаратний інтерфейс. Хоча такий підхід забезпечує високу гнучкість, він має істотний недолік. Кожна операція вводу/виводу проходить через декілька шарів абстракцій, що створює додаткові VM-exit події, збільшує затримки та знижує пропускну здатність. Дослідження вказують, що обробка переривань та повторне копіювання пам’яті у цих архітектурах є ключовим вузьким місцем, що обмежує масштабованість платформи.

На противагу цьому, апаратні VNIO-механізми, такі як SR-IOV, дозволяють гостьовим віртуальним машинам отримувати доступ до фізичного мережевого інтерфейсу через віртуальні функції без повного проходження через гіпервізор. Такий підхід забезпечує майже «bare-metal» швидкодію, проте позбавляє віртуалізовану платформу низки важливих можливостей, зокрема гнучкого керування пам’яттю, міграції VM без переривання роботи та централізованого контролю доступу. Крім того, апаратні рішення відкривають додатковий вектор атаки - канал прямого доступу до пам’яті, який може стати точкою проникнення у випадку компрометації драйвера або гостьової системи.

Найбільш поширеним та збалансованим підходом у сучасних хмарних інфраструктурах є пара-віртуалізований VNIO, зокрема механізми на основі стандарту virtio. Саме цей підхід забезпечує компроміс між високою продуктивністю та функціональною гнучкістю. Пара-віртуалізація дозволяє суттєво скоротити кількість копіювань пам’яті шляхом використання спільних буферів між VM та бекендом, який виконує фактичне передавання пакетів. У моделі virtio обробка пакетів розділена між фронтенд-драйвером у гостьовій ОС та бекенд-складовою на стороні хоста, що дозволяє уникнути повної емуляції циклу.

З часом архітектури пара-віртуалізованого вводу/виводу еволюціонували від реалізацій, де бекенд виконувався в межах QEMU-процесу, до моделей, у яких бекенд переміщено до ядра Linux (vhost-net), і зрештою - до механізмів, у яких бекенд реалізується у користувацькому просторі високопродуктивного комутатора, такого як OVS-DPDK (vhost-user) [20]. Саме остання модель продемонструвала найвищу ефективність, оскільки усуває необхідність у переходах між режимами роботи ядра та користувацького простору та дозволяє використовувати високошвидкісні драйвери bypass-типу.

У такій архітектурі vSwitch стає активним учасником обробки пакетів другого рівня та ініціатором доступу до пам'яті VM. Через це об'єктами уваги стають два типи пам'яті, що взаємодіють у механізмі VNIO: буфери віртуальної машини, які містять пакети, що надходять або надсилаються гостьовою ОС, та буфери хоста, які управляються vSwitch і використовуються як проміжні сховища для пакетів, що обробляються. На рисунку 2.3 наведено схему типової взаємодії між VM, гіпервізором, vSwitch та фізичним NIC.

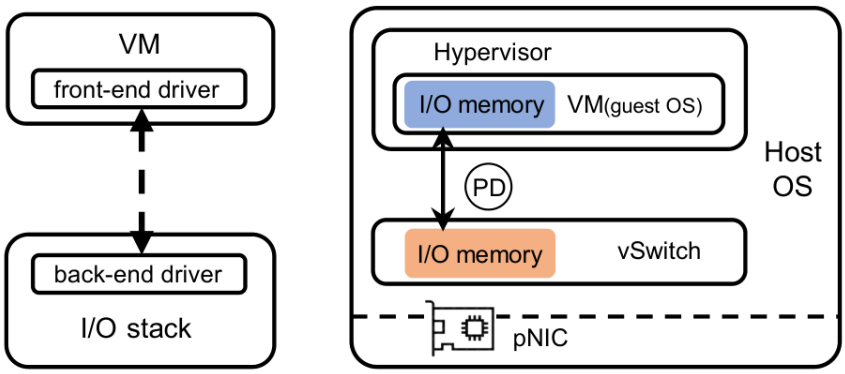



Рисунок 2.2 – Схема взаємодії між VM, гіпервізором, vSwitch та фізичним NIC

Тут видно, що віртуальна машина взаємодіє з бекендом через структури virtqueue, а vSwitch, виконуючи функції приймання і переспрямування, працює безпосередньо з хостовими буферами, які виступають тимчасовим сховищем трафіку.

Архітектура пара-віртуалізованих систем вводить принцип спільного доступу до пам'яті, що використовується для оптимізації операцій копіювання

та мінімізації системних переходів. Саме ця властивість дозволяє добитися продуктивності, порівнянної з апаратними прискорювачами, проте водночас породжує ризики порушення ізоляції між компонентами. Спільна пам’ять може бути реалізована як відображення пам’яті VM у vSwitch або як відображення хостового буфера у гостьову систему, залежно від конкретного механізму. У пара-віртуалізованих рішеннях попередніх поколінь, таких як virtio-net та vhost-net, питання ізоляції не були критичними, оскільки бекенд функціонував у привілейованому середовищі, яке мало право доступу до пам’яті VM за замовчуванням. Проблеми почали виникати у моделях користувацького простору, де vSwitch більше не має привілейованого статусу, але для забезпечення високої швидкодії все одно отримує доступ до пам’яті всіх VM.

Централізований програмнний комутатор, який працює у просторі користувача, також вплинула на форму взаємодії між VNIO та мережевою інфраструктурою хоста. Багатопотокові механізми обробки пакетів, такі як PMD, дозволяють vSwitch працювати в режимі безперервного опитування буферів NIC, уникаючи системних викликів та переривань. Така взаємодія суттєво знижує накладні витрати, збільшує пропускну здатність і забезпечує передбачуваний час обробки. У поєднанні з механізмами DPDK та пулами пам’яті типу mbuf це створює високошвидкісний конвеєр обробки пакетів між фізичним інтерфейсом та віртуальними машинами.

Вся архітектура VNIO функціонує над базовим компонентом - гіпервізором, який реалізує логіку створення, виконання та обслуговування VM. У середовищах на основі QEMU/KVM гіпервізор складається з модулю KVM у ядрі Linux та окремих QEMU-процесів, кожен з яких відповідає за конкретну віртуальну машину. При цьому QEMU відповідає за створення віртуальних процесорів (vCPU), ініціалізацію пам’яті гостьової ОС, емуляцію пристроїв та взаємодію з бекендом VNIO. З огляду на це гіпервізор має повний і легітимний доступ до пам’яті VM, що є важливим фактором при розгляді параметрів ізоляції, проте на етапі первинного аналізу архітектури такі особливості слід розглядати лише як складову загальної взаємодії компонентів, а не як основу для подальших моделей безпеки. Загалом архітектурні основи віртуалізованих мережевих

підсистем у сучасних хмарних середовищах формують складну багаторівневу структуру, у якій гостьові ОС, гіпервізор, програмний комутатор і механізми VNIO взаємодіють у щільно інтегрованому конвеєрі обробки пакетів. Відмінності між підходами до обробки мережевого вводу/виводу, а також еволюція пара-віртуалізації від virtio-net до vhost-user визначили сучасний стан індустрії, у якому продуктивність досягає рівня, близького до апаратного, але виникає потреба у значно більш жорстких гарантіях ізоляції пам’яті. Саме ці базові архітектурні засади створюють передумови для формулювання проблематики, що розглядається у подальших підрозділах цього розділу.

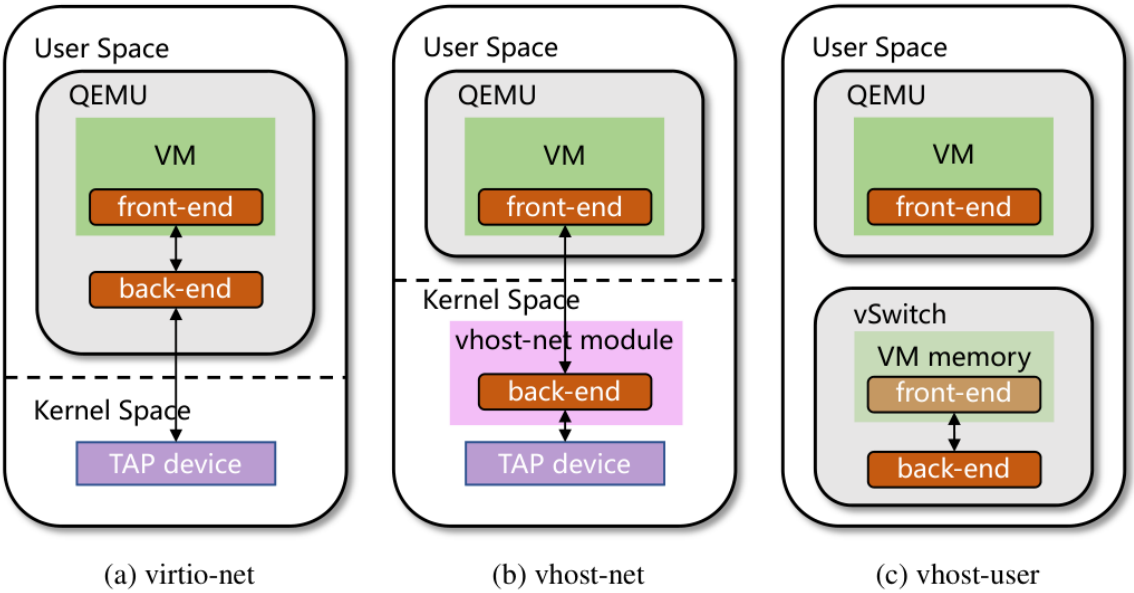
## 2.2 Еволюція пара-віртуалізованого VNIO

Еволюція пара-віртуалізованих механізмів VNIO у сучасних хмарних платформах є результатом багаторічного пошуку оптимального компромісу між продуктивністю, універсальністю та рівнем ізоляції між віртуальними машинами. Від перших, відносно простих рішень до високопродуктивних механізмів користувацького простору, технології VNIO поступово вдосконалювалися, щоб подолати обмеження повної віртуалізації та максимально наблизити швидкодію до фізичного обладнання. Водночас, із кожною фазою розвитку зростала складність механізмів доступу до пам’яті, що згодом стало одним із центральних викликів безпеки віртуалізованих середовищ.

Витоки пара-віртуалізації пов’язують із проблемою надмірних накладних витрат повної емуляції мережевих пристроїв гіпервізором. У класичних реалізаціях гіпервізор перехоплював кожну операцію вводу/виводу, генеруючи VM-exit події та передаючи управління до свого процесу для обслуговування запиту. При цьому кожна операція супроводжувалася копіюванням буферів, перевірками та емуляцією поведінки фізичного NIC. Такий підхід суттєво обмежував пропускну здатність і масштабованість, особливо в сценаріях багатокористувацьких хмарних середовищ. Тому ще у 2005 році IBM запропонувала стандарт virtio, який започаткував епоху пара-віртуалізованого вводу/виводу. Його основна ідея полягала у створенні двоскладової моделі, що

поєднує фронтенд у гостьовій системі та бекенд на стороні хоста. Фронтенд виконував роль драйвера спрощеного віртуального пристрою, тоді як бекенд реалізовував фактичні операції доступу до мережевого інтерфейсу.

На рисунку 2.3 схематично зображено три покоління механізмів virtio, які відображають ключові етапи еволюції VNIO.




(a) virtio-net (b) vhost-net (c) vhost-user

Рисунок 2.3 – Три покоління механізмів virtio

На першій фазі, представлено варіант virtio-net, у якому бекенд виконаних функцій інтегровано у процес QEMU. Саме цей процес обслуговував черги virtqueue, копіював пакети до TUN/TAP-пристрою та здійснював обробку переривань [21]. Така організація мала суттєвий недолік: гіпервізорний процес QEMU, відповідальний за управління пам’яттю VM, був змушений додатково виконувати інтенсивні операції мережевого вводу/виводу. Це призводило до перевантаження QEMU, появи затримок, непередбачуваних пауз у роботі VM та зниження загальної стабільності системи у разі високого трафіку.

Наступна фаза розвитку полягала у перенесенні бекенд-функціоналу з простору QEMU до ядра операційної системи. Механізм vhost-net, підтримує обробку пакетів безпосередньо у ядрі Linux. Завдяки цьому виключено значну частку операцій контекстного перемикання та копіювання між користувацьким простором та простором ядра. Крім того, ядро має прямий доступ до пам’яті

гостьових машин через функціонал гіпервізора (зокрема через механізми KVM та memory-mapping), тому розміщення бекенду в просторі ядра дозволяло значно покращити продуктивність передачі пакетів. У моделі vhost-net саме ядро відповідало за керування буферами, доступ до NIC та обробку черг virtqueue, що дозволило досягти істотного прискорення шляхом використання мінімального шляху доставки пакетів без участі QEMU у кожній транзакції.

Попри це, розвиток апаратного забезпечення, поява багатоядерних процесорів і поширення технологій на кшталт DPDK сприяли зростанню нового напряму - мережевої обробки у просторі користувача. Цей підхід дозволив повністю узгодити кешування даних, перепланувати використання апаратних черг NIC, мінімізувати системні виклики та реалізувати високопродуктивні PMD-потоки, що працюють у режимі активного опитування без переривань. Ключову роль у цій трансформації відіграли програмні комутатори нового покоління, такі як OVS-DPDK, які інтегрували механізми швидкісного доступу до NIC та обробку фреймів другого рівня на рівні користувача. Третє покоління virtio, позначене як vhost-user, стало логічним продовженням цієї тенденції. У моделі vhost-user бекенд функціонує у просторі користувача, всередині vSwitch, який працює в DPDK-середовищі. Таким чином, практично всі операції обробки, класифікації та переспрямування пакетів виконуються в користувацькому просторі з максимально можливою швидкістю. Щоб реалізувати такий механізм, vSwitch повинен отримати доступ до пам’яті віртуальних машин, у яких розміщуються буфери virtio. Оскільки програмний комутатор не може працювати з привілейованим кодом, цей доступ організовують через масивні операції mmap(), які відображають пам’ять VM у адресу vSwitch. Саме це дозволяє vSwitch обходити ядро і напряму читати або записувати пакети, зберігаючи максимальну пропускну здатність і низькі затримки у напрямах від vNIC до pNIC та в зворотному напрямку.

Еволюція VNIO проходила в умовах постійного зростання вимог до хмарних платформ, які мають підтримувати величезну кількість VM на одному фізичному сервері. Така щільність розміщення ресурсів вимагала розроблення систем, здатних ефективно використовувати CPU, пам’ять та мережеві ресурси.

Саме тому пара-віртуалізація стала фактично стандартом для виробників інфраструктури, включно з Google, Alibaba Cloud та іншими провайдерами, які впроваджують власні модифікації virtio та vhost-механізмів у масштабованих середовищах.

Особливість третього покоління VNIO полягає не лише в тому, що бекенд переноситься до vSwitch, а й у тому, що саме vSwitch стає центральною точкою прийняття рішень. Його завдання не обмежуються передаванням пакетів. Він виконує класифікацію трафіку, взаємодію з таблицями потоків, реалізує політики контролю доступу та QoS. Щоб досягти цього без зниження продуктивності, vSwitch має працювати у тісному зв’язку з NIC і пам’яттю VM. Така синергія дозволяє створити конвеєр, у якому копіювання відбувається мінімально можливою кількістю разів.

Однак такий підхід має й іншу сторону. Модель vhost-user, хоч і забезпечує максимальну продуктивність, створює передумови для серйозних викликів у сфері безпеки. Відображення пам’яті VM у vSwitch означає, що процес, який виконуються у просторі користувача, отримує доступ до повної пам’яті орендарів. Це порушує традиційний принцип ізоляції, який довгий час вважався базовою гарантією віртуалізованих платформ. Механізм mmap-проекцій дозволяє vSwitch працювати продуктивно, але фактично робить пам’ять VM доступною будь-якому коду, що працює у процесі vSwitch, включно з потенційно вразливими або зламаними компонентами.

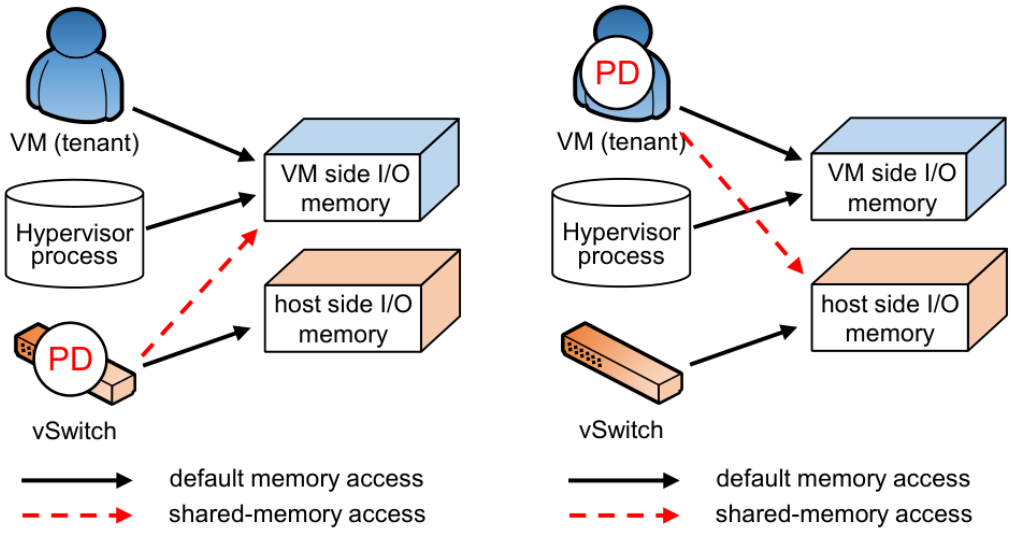
Таким чином, еволюція пара-віртуалізованих VNIO від virtio-net через vhost-net до vhost-user демонструє послідовний перехід від моделей, орієнтованих переважно на сумісність та безпеку, до моделей, які ставлять продуктивність на перший план. Хоча кожне покоління усувало недоліки попереднього, воно одночасно породжувало нові проблеми, пов’язані з балансуванням між швидкодією і гарантіями ізоляції. На момент появи vhost-user цей баланс суттєво змістився у бік продуктивності. Подальший розвиток індустрії засвідчив, що максимальне наближення до bare-metal швидкодії шляхом перенесення бекенду до vSwitch повинно супроводжуватися

переосмисленням моделей ізоляції і пошуком нових підходів, які змогли б поєднати переваги попередніх поколінь без відмови від продуктивності.

Саме розуміння цього еволюційного шляху та його наслідків формує основу для аналізу проблематики ізоляції пам’яті, яка розглядається у наступних підрозділах цього розділу.

## 2.3 Моделі обміну пам’яттю у пара-віртуалізованих VNIO

Пара-віртуалізований мережевий ввід/вивід у сучасних хмарних платформах ґрунтується на тісній взаємодії між віртуальною машиною, гіпервізором і vSwitch, які мають спільно забезпечити високу пропускну здатність та мінімальні затримки під час передавання пакетів. Однак саме ця взаємодія визначає характер обміну пам’яттю між компонентами й формує ризики, які напряму впливають на ізоляцію орендарів та безпеку платформи. Однією з ключових особливостей пара-віртуалізованих VNIO-механізмів є залежність від спільної пам’яті, яка дозволяє скоротити кількість копіювань пакетів і зменшити накладні витрати. З іншого боку, необхідність відображення пам’яті між компонентами створює фундаментальні виклики щодо контролю доступу, оскільки розмиваються межі між пам’яттю VM та пам’яттю хоста. На рисунку 2.4 представлено два принципово різних підходи до організації спільної пам’яті: модель VM2vS, у якій віртуальна машина надає доступ до своєї пам’яті програмному комутатору, та модель vS2VM, у якій vSwitch надає спільний блок пам’яті віртуальній машині.




(а) Model VM2vS (б) Model vS2VM

Рисунок 2.4 – Підходи до організації спільної пам’яті

Ці дві моделі виникли в результаті пошуку способів забезпечити мінімальні затримки при виконанні PD - процесу передавання та копіювання мережевих даних між компонентами. Обидві моделі потребують, щоб PD мав доступ до двох сторін: пам’яті VM, де розташовані буфери virtqueue, та пам’яті хоста, де vSwitch зберігає власні буфери для обробки трафіку. Обирати модель обміну пам’яттю доводиться відповідно до того, де виконується PD-процедура.

Модель VM2vS бере свій початок з підходів, реалізованих у virtio vhost-user, Google Andromeda та інших промислових системах. У цьому випадку vSwitch працює у просторі користувача та виконує функції PD. Щоб він мав можливість копіювати дані між власними хостовими буферами та буферами VM, пам’ять гостьової ОС відображається всередині vSwitch через низку системних викликів mmap(). Таким чином, vSwitch отримує можливість доступу до цих ділянок пам’яті напряму, без залучення ядра. У свою чергу гіпервізор та VM мають природний доступ до пам’яті віртуальної машини, що зберігає узгодженість між компонентами. Така модель виявилася ефективною з точки зору продуктивності, оскільки забезпечує прямий доступ користувацького vSwitch до буферів virtqueue, що дозволяє реалізувати найкоротший можливий шлях транспортування пакетів.

Однак модель VM2vS поступово перетворилася на джерело проблем безпеки. Основна причина полягає в тому, що vSwitch  незважаючи на свою важливість є процесом користувацького простору, у якому виконується складна, велика за кодовою базою та потенційно вразлива логіка. Розміщення vSwitch у user space дозволяє застосовувати високопродуктивні механізми на кшталт DPDK, але позбавляє його привілейованого статусу, який раніше мали kernel-based реалізації. Наслідком цього є те, що відображення пам’яті VM у адресу vSwitch фактично передає процесу доступ до всієї пам’яті орендаря, а зазвичай навіть до пам’яті всіх VM на даному хості, оскільки під час ініціалізації vhost-user неможливо деталізувати точний діапазон I/O-пам’яті, який буде використовуватися virtio. Таким чином, для забезпечення працездатності VNIO vSwitch отримує повністю довірений доступ до ділянок пам’яті, які не повинні бути доступні користувацькому процесу за жодних умов. У разі компрометації vSwitch будь-яким вразливим модулем або експлойтом зловмисник отримує можливість зчитувати й змінювати пам’ять інших VM, що руйнує модель безпеки багатокористувацької хмари.

Інша модель виникла в альтернативних VNIO-проєктах, таких як NetVM, IVSHMEM та ClickOS. У цих системах PD-процедура виконується у гостьовій VM. Для цього vSwitch виділяє окремий блок пам’яті  так званий host-side I/O memory  і передає його до VM через механізм спільного доступу, що імітує роботу PCIe-пристрою. Віртуальні машини отримують доступ до цього блоку через vPCI, і PD виконується шляхом читання та запису в спільні буфери. Цей підхід дозволяє vSwitch не мати доступу до пам’яті VM, що створює цікаву асиметрію у порівнянні з VM2vS. Він також зберігає високу продуктивність, адже копіювання пакетів може здійснюватися напряму з NIC до спільної пам’яті без зайвих транзакцій.

Попри це, модель vS2VM має інший клас проблем. Спільна пам’ять у цьому випадку стає доступною гостьовій операційній системі, яка може бути скомпрометована вже самим орендарем або сторонніми атаками. Оскільки у цій моделі PD виконується в межах VM, потенційний зловмисник, що отримав контроль над гостьовою ОС, має можливість маніпулювати спільною пам’яттю,

зчитувати пакети інших орендарів або навіть впливати на функціонування vSwitch, якщо реалізація vPCI-пристрою має недоліки. Складність драйверної логіки у VM робить такі реалізації вразливими, і саме ця причина стала фактором струмування для широкого розповсюдження vS2VM у комерційних хмарних платформах.

Практично всі високопродуктивні VNIO-рішення покладаються на спільну пам’ять, що й робить питання ізоляції центральним викликом для VNIO нового покоління. Обидві моделі VM2vS і vS2VM дозволяють скоротити кількість переходів між ядром і користувацьким простором, але роблять це ціною надзвичайно високого рівня довіри до компонента, який не повинен мати повний доступ до пам’яті іншого. Для VM2vS таким компонентом є vSwitch, для vS2VM - сама VM. У результаті жодна з моделей не гарантує захисту від атак типу DMA, або маніпулювання спільними буферами.

Спільною рисою обох моделей є те, що вони розглядають пам’ять як єдиний ресурс, який розподіляється між учасниками системи без додаткових прошарків перевірки адресної легітимності. Така архітектурна простота забезпечила VNIO високий рівень продуктивності, але водночас створила структурні слабкості в системах транспортування пакетів. Оскільки PD-процедура потребує миттєвого доступу до адрес буферів, розробники обирали найпростішу схему - пряму проєкцію пам’яті, яка дає доступ до всього простору даних, що використовує virtio. Це забезпечувало відмінні швидкісні характеристики, але послаблювало бар’єр між орендарями, залишаючи платформу вразливою до зловмисних дій користувацьких процесів.

У підсумку можна стверджувати, що дві сформовані моделі VM2vS та vS2VM стали наслідком різних підходів до оптимізації VNIO, але не враховують необхідності створення чіткої ізоляції між компонентами. Їхня еволюція дозволила досягти продуктивності, яка є практично необхідною для сучасних хмарних мереж, проте одночасно створила фундаментальну загрозу безпеці. Саме ця проблема породжує потребу переосмислення моделей та пошуку архітектур, які здатні підтримувати пара-віртуалізований I/O без відмови від ізоляції, що розглядається у подальших підрозділах.

## 2.4 Обмеження існуючих підходів до підсилення безпеки

Проблема ізоляції пам’яті у пара-віртуалізованих механізмах VNIO є наслідком того, як еволюціонувала архітектура передачі пакетів між віртуальними машинами та програмним комутатором. Зі зміщенням критичних операцій із ядра в користувацький простір розробники прагнули досягти максимальної продуктивності шляхом усунення VM-exit, скорочення кількості копіювань та переходів між привілейованими режимами. Проте збільшення швидкодії виявилося нерозривно пов’язаним зі зростанням довіри до компонентів, які за своєю природою не мають бути привілейованими. Саме це стало фундаментальною причиною того, що традиційні підходи до підсилення безпеки в існуючих VNIO-механізмах виявилися неефективними або такими, що несумісні з вимогами високопродуктивних датацентрів.

У моделі VM2vS, що відповідає поширеним реалізаціям типу virtio vhost-user, vSwitch отримує прямий доступ до пам’яті віртуальної машини через механізм відображення сторінок за допомогою mmap(). Оскільки адресний простір для I/O-буферів у VM формується динамічно, у реальних системах vSwitch зазвичай отримує доступ не до окремих буферів, а до майже всього простору пам’яті VM. У результаті процес vSwitch, що виконується у user space і містить значну кількість коду та зовнішніх залежностей, фактично стає компонентом із неконтрольовано високими привілеями. Будь-яка вразливість у стеку vSwitch відкриває шлях до компрометації пам’яті всіх VM, що використовують цей механізм. Для усунення цієї проблеми було запропоновано механізм vIOMMU, який додає проміжний контур адресної трансляції всередині QEMU й тим самим контролює всі запити vSwitch до пам’яті VM. Його впровадження передбачає перенаправлення кожної операції читання або запису через гіпервізорний процес, що дає змогу перевіряти легітимність сторінки та блокувати небезпечні доступи. Логіка vIOMMU на концептуальному рівні здатна нейтралізувати DMA-атаки, але архітектурно виявляється несумісною з високошвидкісним VNIO. Значні затримки, спричинені міжпроцесною

комунікацією, синхронізацією черг і контекстними переходами між QEMU та vSwitch, призводять до різкого падіння пропускної здатності. Це робить механізм практично неприйнятним для хмарних провайдерів, для яких продуктивність VNIO є критичною метрикою.

Інший напрямок роботи над підсиленням безпеки зосереджувався на відмові від користувацького простору в обробці даних і переміщенні PD-операцій назад у ядро операційної системи. У проєктах на кшталт Hyper-switch або Zcopy-vhost PD виконується в kernel space, де існують механізми жорсткішого контролю доступів через розширені таблиці сторінок (EPT) та інші інструменти безпеки ядра. Цей підхід справді забезпечує більш жорстку ізоляцію, однак має суттєвий недолік. Продуктивність пакетної обробки в ядрі навіть у поєднанні з сучасними технологіями XDP або eBPF істотно поступається user-space data-plane на базі DPDK. Результати численних досліджень показують, що повернення PD у ядро фактично нівелює ті переваги, які пара-віртуалізація накопичувала протягом останнього десятиліття. Таким чином, навіть якщо kernel-based рішення підвищують безпеку, вони не відповідають вимогам до продуктивності та масштабування.

Модель vS2VM, яка реалізована в таких системах, як NetVM, IVSHMEM та ClickOS, пропонує концептуально протилежний підхід. Не vSwitch отримує доступ до пам’яті VM, а VM отримує доступ до спільної пам’яті, виділеної vSwitch. Такий підхід усуває ризик, притаманний VM2vS, але створює новий. Оскільки PD у цьому варіанті виконується всередині гостьової ОС, будь-яка компрометація VM автоматично надає зловмиснику контроль над спільною пам’яттю та дає змогу перехоплювати, модифікувати або блокувати трафік інших орендарів. Унаслідок цього модель vS2VM хоч і усуває проблему неконтрольованого доступу з боку vSwitch, але переносить точку вразливості на VM, де хмарний провайдер зазвичай має менший контроль над конфігурацією, оновленнями та програмним середовищем.

Спроби використати апаратні засоби ізоляції пам’яті, такі як Intel SGX або ARM TrustZone, також не призвели до універсального рішення. SGX забезпечує найвищий рівень криптографічного захисту, проте не передбачає використання

у високошвидкісних системах мережевої обробки. Накладні витрати на шифрування, відсутність підтримки великих обсягів даних та різке падіння пропускної здатності (приблизно до 10% від базового показника) роблять SGX повністю непридатним для VNIO у продуктивних датацентрах. Аналогічні обмеження притаманні TrustZone, де необхідність перемикання між світами та обмеження апаратних інструкцій створюють додаткову затримку на критичному шляху обробки пакетів.

Усі перелічені підходи мають спільний недолік. Вони намагаються підсилити безпеку, не змінюючи фундаментальної архітектури спільного доступу до пам’яті між компонентами VNIO. Інакше кажучи, удосконалюються механізми контролю в рамках моделей VM2vS та vS2VM, але не усувається сам факт того, що один з небезпечних компонентів - процес у user space або гостьова машина  отримує безпосередній доступ до чужої пам’яті. Саме цей фактор формує головну перешкоду для захисту. Поверхня атаки зростає пропорційно кількості компонентів, які беруть участь у спільній роботі з пам’яттю.

З огляду на вищесказане, можна зробити висновок, що існуючі підходи підсилення безпеки мають фундаментальну архітектурну обмеженість. Вони дозволяють частково зменшити вплив окремих вразливостей, але не спроможні забезпечити гарантовану ізоляцію пам’яті без істотної втрати продуктивності. Подолання цього конфлікту потребує переосмислення ролі компонентів у VNIO-ланцюгу з виокремленням такого елемента, який має природний привілейований доступ до пам’яті VM і водночас не є потенційно небезпечним користувацьким процесом. Такий підхід, як буде розглянуто у наступних частинах роботи, здатен забезпечити одночасно і продуктивність, і безпеку без вад, характерних для моделей VM2vS та vS2VM.

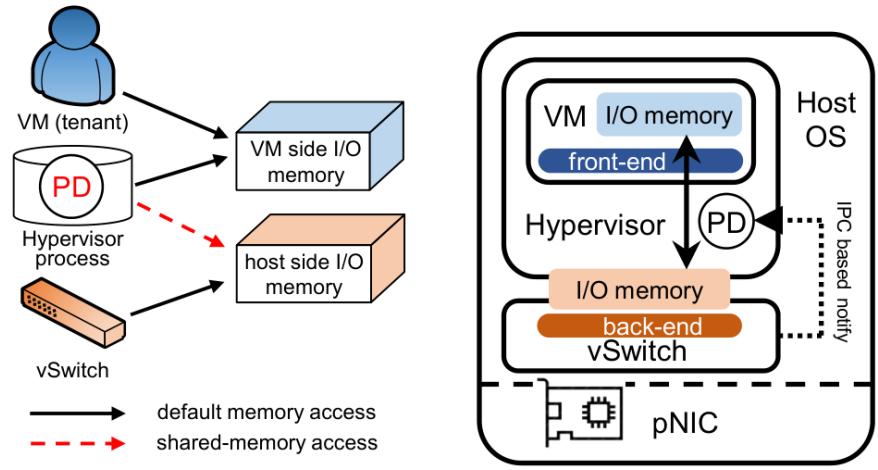
### 2.5 Модель безпечного обміну пам’яттю vSwitch та гіпервізор

Побудова безпечної та високопродуктивної архітектури пара-віртуалізованого мережевого вводу/виводу потребує усунення фундаментального конфлікту між продуктивністю та ізоляцією пам’яті, який

закладений у моделі VM2vS та vS2VM. Обидва підходи, як було показано попередньо, допускають ситуацію, коли небезпечний або потенційно вразливий компонент отримує прямий доступ до чужого адресного простору. Модель VM2vS відкриває доступ процесу vSwitch до пам’яті VM, тоді як vS2VM робить доступною для VM пам’ять хоста, що використовується для зберігання I/O-буферів. Це створює передумови як для атак на віртуалізаційну платформу. З огляду на це було запропоновано нову модель безпечного обміну пам’яттю - vS2H, яка перекладає виконання механізмів PD на гіпервізор і повністю усуває необхідність прямого доступу VM до пам’яті vSwitch або vSwitch до пам’яті VM.

Архітектурно модель vS2H ґрунтується на простому та водночас фундаментальному принципі. Єдиним компонентом системи, що має законний, контрольований і технічно необхідний доступ до пам’яті віртуальної машини, є гіпервізор. Саме він керує життєвим циклом VM, ініціалізує її адресний простір, відстежує сторінки пам’яті, обробляє інструкції змішаного рівня привілеїв і забезпечує виконання віртуальних CPU. Гіпервізор має повні привілеї не тому, що це бажано з точки зору функціональності, а тому, що без цього він не може виконувати базові функції віртуалізації. Тому будь-який механізм роботи з пам’яттю, що покладається на гіпервізор, автоматично отримує системні гарантії безпеки, притаманні всій платформі.

Центральна ідея моделі vS2H полягає в тому, що гіпервізор виконує функцію посередника між пам’яттю VM та пам’яттю vSwitch, який розташований у user space. На рисунку 2.5 подано концептуальну схему цієї моделі.




(а) Model vS2H (б) Архітектура vS2H

Рисунок 2.5 – Концептуальна схема vS2H

На відміну від VM2vS, де vSwitch працює з пам’яттю VM, та vS2VM, де VM працює з пам’яттю vSwitch, у vS2H ці два компоненти не мають жодного прямого доступу один до одного. Гіпервізор отримує від vSwitch блок I/O-пам’яті лише той, який необхідний для передачі пакетів і додатково використовує власний привілейований доступ до пам’яті VM. Це дозволяє виконувати PD-процедуру всередині гіпервізора, не відкриваючи поверхню атаки для інших компонентів.

Розширена архітектура демонструє цей принцип у контексті стандартної пара-віртуалізаційної моделі virtio. У традиційній VM2vS реалізації vhost-user PD-виконання відбувається всередині vSwitch, тоді як у vS2H PD-потік запускається всередині процесу гіпервізора (наприклад, QEMU у випадку KVM). Гіпервізор читає дескриптори пакетів із виділеної області пам’яті vSwitch, копіює необхідні дані у відповідні буфери VM і оновлює virtqueue так, як це робив би vSwitch. Зворотний напрям руху пакетів виконується аналогічно. Такий підхід дозволяє повністю зберегти сумісність із virtio, оскільки поведінка віртуального інтерфейсу змінюється лише з погляду розташування PD-процедури, але не з погляду віртуальної машини.

Суттєвою перевагою vS2H є те, що він усуває необхідність відображати пам’ять VM у просторі vSwitch, що було основним джерелом ризику у VM2vS. Водночас ця модель не потребує доступу VM до пам’яті хоста, який був необхідним для vS2VM. В обох попередніх випадках саме спільна пам’ять

порушувала межі ізоляції, що й дозволяло здійснювати атаки типу DMA або маніпуляції з неконтрольованими буферами. У vS2H спільна пам’ять існує лише між vSwitch та гіпервізором, але ця пам’ять є строго обмеженою у розмірі та містить лише I/O-дані, необхідні для PD-процедури. Пам’ять VM нікуди не відображається, що повністю блокує можливість неконтрольованого доступу.

Архітектурна чистота моделі vS2H дозволяє усунути ще один критичний недолік класичних підходів - проблему визначення та обмеження I/O-діапазонів пам’яті. У VM2vS vSwitch має доступ до всієї пам’яті VM, оскільки неможливо відстежити динамічні зміни virtqueue та адрес буферів. У vS2VM гостьова ОС отримує доступ до великого статичного блоку, що постійно перебуває у спільному просторі пам’яті. У vS2H обидві області суворо визначені наперед: гіпервізор знає точний розмір і структуру пам’яті VM, а блок I/O-пам’яті vSwitch є фіксованим та контрольованим. Це усуває можливість випадкового або навмисного виходу за межі дозволених адрес.

Важливою рисою моделі vS2H є те, що вона зберігає продуктивність, характерну для пара-віртуалізації, без потреби повернення PD до kernel space. На відміну від рішень, які намагаються підвищити безпеку шляхом перенесення I/O-операцій у ядро, vS2H залишає всю інтенсивну обробку в user space vSwitch, але відбирає в нього критичну частину - прямий доступ до пам’яті VM. Такий поділ дозволяє vSwitch продовжувати використовувати високопродуктивні механізми на кшталт DPDK, зберігаючи більшу частину пропускної здатності vhost-user, але не має ризиків, пов'язаних із пам'ятною ізоляцією.

Розміщення PD у гіпервізорі має ще одну важливу властивість. Можливість ізолювати обробку пакетів для кожної VM у межах її власного процесу гіпервізора. Це дозволяє забезпечити фізичну сегментацію потоків обробки, унеможливлюючи витік інформації між VM навіть на рівні кеш-пам’яті або внутрішніх структур vSwitch. Хоча цей аспект виходить за межі класичного поняття ізоляції пам’яті, він безпосередньо впливає на загальну модель безпеки VNIO, оскільки зменшує кількість побічних каналів передачі інформації.

Модель vS2H є логічним завершенням аналізу недоліків VM2vS та vS2VM. Вона не просто відмежовує компоненти один від одного, а переводить критичний

механізм PD у єдиний компонент, який вже має необхідний рівень довіри та контролю - гіпервізор. Такий підхід усуває причини вразливостей, пов'язаних із неконтрольованим доступом до спільної пам’яті, і пропонує архітектуру, у якій пара-віртуалізований VNIO може забезпечувати ізоляцію пам’яті без втрати продуктивності. Таким чином модель vS2H створює фундамент для побудови безпечних, масштабованих і продуктивних мережевих підсистем у сучасних хмарних віртуалізованих середовищах.

## 2.6 Висновки до розділу 2

В другому розділі було виконано комплексний аналіз архітектурних та безпекових аспектів організації віртуалізованого мережевого вводу/виводу у хмарних середовищах. Порівняння повної віртуалізації, апаратних рішень та пара-віртуалізації продемонструвало, що саме пара-віртуалізований підхід на основі virtio забезпечує найкращий компроміс між гнучкістю й продуктивністю, але одночасно переносить критичне навантаження на механізми спільного доступу до пам’яті.

У межах розділу було формалізовано дві базові моделі обміну пам’яттю у пара-віртуалізованих VNIO: VM2vS, де пам’ять VM відображається у vSwitch, та vS2VM, де vSwitch надає спільну I/O-пам’ять віртуальним машинам. Показано, що в моделі VM2vS вразливим елементом стає vSwitch, який у випадку компрометації отримує можливість неконтрольованого доступу до пам’яті орендарів. У моделі vS2VM точка вразливості зміщується до VM, яка, маючи доступ до спільних буферів, може впливати на трафік інших віртуальних машин. В обох випадках сам принцип прямого спільного доступу до пам’яті між недовіреними або частково довіреними компонентами створює глибинні структурні проблеми для ізоляції.

З урахуванням виявлених обмежень було обґрунтовано формування нової моделі безпечного обміну пам’яттю vS2H, у якій критичні функції PD переносяться до гіпервізора. У цій моделі гіпервізор виступає єдиним посередником між пам’яттю VM та I/O-пам’яттю vSwitch, тоді як ні VM, ні

vSwitch не мають прямого доступу до пам’яті одне одного. Такий перерозподіл ролей спирається на той факт, що гіпервізор уже є привілейованим і контрольованим компонентом із повним доступом до пам’яті VM, і саме він є природним кандидатом для виконання PD без розширення поверхні атаки. Модель vS2H дозволяє зберегти основні переваги пара-віртуалізованого VNIO, наближені до продуктивності vhost-user, але при цьому кардинально посилює ізоляцію пам’яті та мінімізує ризик атак.

Узагальнюючи результати другого розділу, можна стверджувати, що проведений аналіз висвітлив системну природу проблеми ізоляції у віртуалізованому мережевому I/O, яка не зводиться до окремих вразливостей чи реалізаційних помилок. Вона є наслідком еволюції архітектур, що ставили продуктивність вище за безпеку пам’яті. Запропонована концепція моделі vS2H формує теоретичне підґрунтя для подальшої розробки механізму VNIO, який одночасно забезпечує високу пропускну здатність і строгі гарантії ізоляції, що й стане предметом детального опрацювання в наступному розділі.

# РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ПРОДУКТИВНОСТІ МЕХАНІЗМУ VS2H

## 3.1 Методика оцінювання продуктивності та експериментальне середовище

Оцінювання ефективності механізму віртуалізованого мережевого введення-виведення на основі моделі vS2H потребує ретельно спроєктованої експериментальної методики, яка дозволяє відокремити власне вплив архітектурних змін на критичні показники пропускної здатності, затримки та стабільності від впливу периферійних факторів, притаманних гіпервізору, віртуальним машинам і стеку мережевих протоколів. Основним завданням експериментів є перевірка того, наскільки перенесення частини функцій обробки пакетів від віртуального комутатора до гіпервізора (що становить сутність моделі vS2H) впливає на швидкодію та передбачуваність роботи віртуалізованої мережевої інфраструктури. У цьому контексті методика побудована так, щоб мінімізувати відмінності між конфігураціями порівнюваних архітектурних рішень, забезпечити повторюваність результатів, ізолювати випадкові флуктуації продуктивності та гарантувати коректність висновків.

В основі експериментальної процедури лежить концепція вимірювання продуктивності віртуалізованого VNIO-шляху, яка пов’язана з безпосереднім аналізом швидкості обробки пакетів у траєкторії між віртуальною машиною та фізичним мережевим інтерфейсом. Оскільки будь-які зміни у механізмах копіювання дескрипторів, обробці сигналів, синхронізації між потоками та організації доступу до пам’яті можуть впливати на кінцеві показники пропускної здатності та затримки, експериментальне середовище побудоване таким чином, щоб відобразити реальні навантаження та сценарії, одночасно забезпечуючи точність вимірювань відповідно до рекомендацій RFC 2544 . Згідно з методикою, throughput та latency вимірюються за умови нульових втрат пакеті [22], що дозволяє порівнювати архітектури в умовах максимальної стабільності роботи та гарантує, що результати не спотворюються ефектами чергування або переповнення буферів.

Для дослідження використано серверну платформу середнього рівня продуктивності, що дозволяє відтворювати реальні сценарії роботи невеликого приватного хмарного середовища чи корпоративної віртуалізованої інфраструктури. Сервер оснащено двома процесорами Intel Xeon E5-2620 v2 із робочою частотою 2.10 GHz, кожен з яких містить шість фізичних ядер із підтримкою двох логічних потоків. Обрана конфігурація є достатньою для виконання дослідження, але водночас забезпечує додаткову релевантність отриманих результатів для середовищ, де доступні апаратні ресурси є обмеженими. Система має 64 ГБ оперативної пам'яті DDR3 1600 MHz (див. рисунок 3.1).

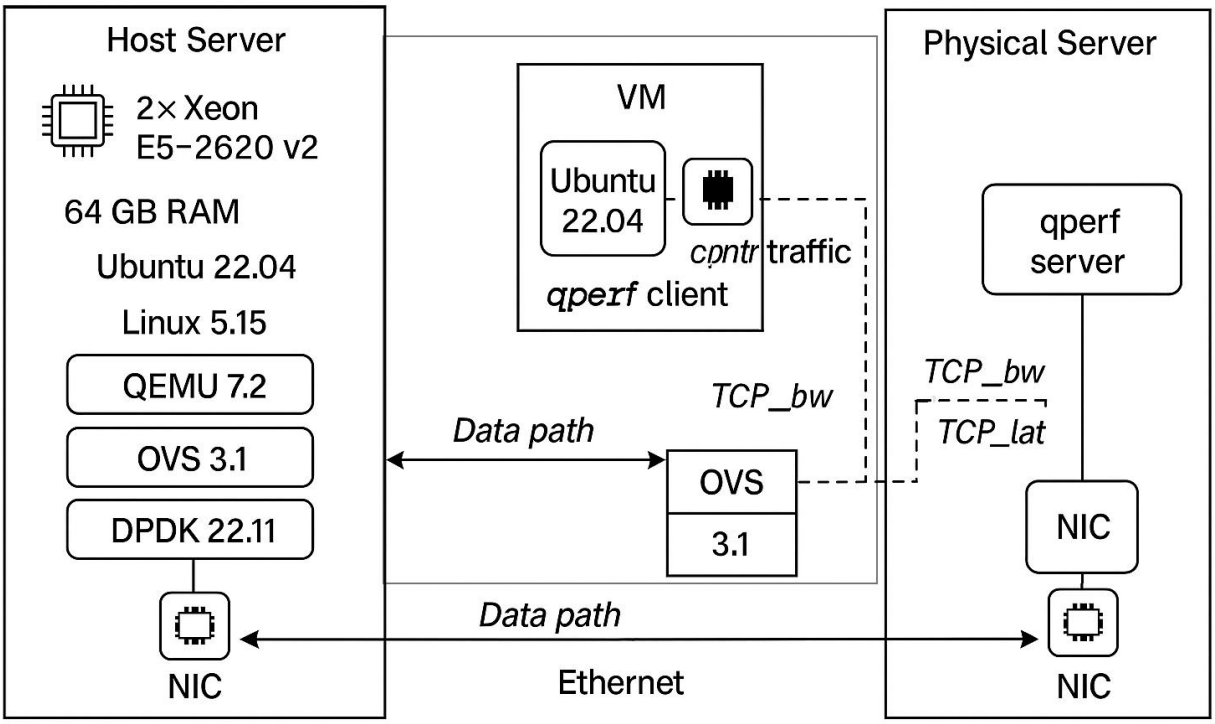



Рисунок 3.1 – Схема тестування

Весь експериментальний стенд працює під керуванням Ubuntu 22.04 LTS із ядром Linux 5.15, причому однакова версія операційної системи використовується як на хості, так і всередині віртуальних машин. Це дозволяє уникнути різниці у поведінці мережевого стеку та гарантує стабільність результатів вимірювань. Підсистема віртуалізації побудована на QEMU 7.2, який забезпечує розширену підтримку апаратної віртуалізації, оптимізовані шляхи

доступу до пам’яті та повну сумісність із сучасними механізмами прискореної обробки пакетів. Віртуальний комутатор реалізовано за допомогою Open vSwitch 3.1, що підтримує як традиційний vHost-User, так і прискорені DPDK-режими. Усі експерименти виконуються з використанням DPDK 22.11 (LTS), який містить оптимізовані драйвери userspace, покращені механізми NUMA-aware memory management та ефективні засоби взаємодії між vSwitch і гостьовою операційною системою. Застосування цих компонентів забезпечує сучасний рівень продуктивності VNIO і дозволяє оцінити поведінку архітектури vS2H в умовах, наближених до реальних хмарних інфраструктур.

Конфігурація віртуальних машин навмисно уніфікована: кожній VM виділяється один логічний процесор і 2 ГБ пам’яті. Такий підхід дозволяє контролювати вплив процесорної конкуренції на результати вимірювань та запобігає ситуації, коли застосунок у VM стає вузьким місцем системи. Усі експерименти повторюються десятикратно, що дає можливість усереднити значення та усунути статистичні викиди, пов’язані з ефемерною активністю фонових служб або випадковими ефектами планування процесів.

Ключовим елементом методики є використання високопродуктивного генератора трафіку TRex Traffic Generator, який працює в режимі користувацького простору на основі DPDK і забезпечує формування мережевих пакетів із точним контролем їхнього розміру, швидкості та інтенсивності [23]. TRex дозволяє генерувати трафік із фіксованими розмірами кадрів - 64, 128, 256, 512, 1024 та 1518 байт, що відповідає діапазону, типовому для експериментальних досліджень віртуалізованих мереж і повністю відтворює сценарії, необхідні для аналізу VNIO-підсистеми. Завдяки роботі поверх DPDK TRex забезпечує стабільне та детерміноване генерування трафіку без участі мережевого стеку ядра Linux, що усуває побічні ефекти, характерні для програмних генераторів, зокрема неконтрольовані блокування, програмні затримки та залежність від планування процесів. Це дозволяє точно вимірювати пропускну здатність, затримку та втрати пакетів у VNIO-шляху навіть при високих навантаженнях, забезпечуючи коректність і відтворюваність експериментальних результатів.

Для оцінювання параметрів TCP-з’єднань застосовано інструмент qperf, який працює на окремому фізичному сервері з ідентичною апаратною конфігурацією [24]. Використання окремого вузла дозволяє уникнути внутрішніх маршрутизаційних петель усередині хостової системи, що часто виникають при локальному тестуванні, та забезпечує коректну характеристику пропускної здатності TCP без впливу оптимізацій ядра, механізмів loopback чи взаємної конкуренції потоків усередині одного фізичного сервера. Такий підхід дає змогу зосередитись саме на поведінці віртуалізованого механізму передачі даних, оскільки вимірювання відображають пропускну здатність і затримку на повному шляху: від віртуальної машини через VNIO-підсистему, віртуальний комутатор, фізичний мережевий інтерфейс і до зовнішнього сервера.

Принцип вимірювання в qperf полягає у тому, що на фізичному сервері запускається qperf у режимі пасивного очікування вхідних з’єднань, тоді як у віртуальній машині працює клієнтська частина, яка ініціює TCP-тестування. Під час оцінювання пропускної здатності (tcp_bw) клієнт посилає потокові дані визначеного розміру упродовж заданого інтервалу часу, а сервер фіксує фактичний обсяг отриманих байтів та розраховує середню пропускну здатність каналу. У режимі вимірювання затримки (tcp_lat) qperf організовує послідовність коротких транзакцій «запит–відповідь», у межах яких кожен пакет супроводжується часовою міткою, що дає змогу точно визначити RTT та оцінити вплив віртуалізованої VNIO-підсистеми на часові характеристики TCP. Таким чином qperf вимірює як однобічну пропускну здатність TCP-каналу, так і часові параметри обміну між клієнтом і сервером, що є особливо важливим для аналізу впливу архітектурних змін на стабільність та реактивність мережевих застосунків.

Особливу увагу в методиці приділено забезпеченню симетрії умов між двома порівнюваними архітектурами. Оскільки у vS2H частина навантаження, пов’язаного з обробкою пакетів, переноситься від PMD-потоку OVS до потоків, які працюють у контексті QEMU, важливо забезпечити однакову кількість виділених логічних ядер для PD-процедур у кожній архітектурі. Саме тому в експериментах використано фіксоване виділення одного логічного ядра для PD-

потоку vHost-User та аналогічної кількості для PD-потоків у vS2H. Для останнього також встановлено мінімальний поріг часу очікування сну, щоб досягти максимальної продуктивності, що відповідає налаштуванням, наведеним у вихідній роботі.

Методика також включає моніторинг стабільності під час тривалих експериментів, що дозволяє переконатися у відсутності деградації продуктивності, спричиненої фрагментацією пам’яті, накопиченням дескрипторів або зміною поведінки потоків через теплові обмеження процесора. Оскільки модель vS2H передбачає використання спільної пам’яті між компонентами системи, важливо гарантувати, що експерименти не зазнають впливу внутрішніх механізмів очищення кешу чи вирівнювання доступу до NUMA-домена. З цією метою всі VM та процеси OVS розміщено всередині одного NUMA-вузла.

### 3.2 Оцінювання продуктивності каналу даних

У процесі дослідження ефективності моделі vSwitch-to-Hypervisor особливе значення має аналіз продуктивності каналу даних, або VNIO datapath, який визначає швидкість і стабільність доставки пакетів між фізичним мережевим інтерфейсом та віртуальними машинами. Саме поведінка datapath є фундаментальною для розуміння переваг чи недоліків будь-якої архітектури віртуалізованого мережевого введення-виведення, оскільки її характеристики безпосередньо впливають на пропускну здатність системи, навантаження на CPU та інтеграцію з віртуальним комутатором. Оцінювання продуктивності datapath дає змогу ізолювати ефект архітектурних змін від впливу протоколів вищого рівня, тому в цьому підрозділі аналіз фокусується на “чистій” передачі Ethernet-фреймів без додаткових обчислювальних операцій з боку стеку ядра або прикладних програм у віртуальній машині.

Вихідною точкою для проведення вимірювань стало застосування високопродуктивного TRex Traffic Generator, який дозволяє формувати потоки пакетів фіксованого розміру та надсилати їх із максимально можливою

швидкістю. Такий підхід є важливим, оскільки забезпечує повноцінне навантаження на VNIO-підсистему й дає змогу оцінити пропускну здатність у граничних умовах. На відміну від джерел трафіку, інтегрованих у програмний стек, TRex працює поверх DPDK і не залежить від алгоритмів планування ядра Linux, що унеможливлює появу додаткових джерел затримок, пов'язаних з обробкою переривань, чергами мережевого драйвера чи обмеженнями TCP/UDP. Саме тому вимірювання datapath у такому середовищі є максимально точними та репрезентативними.

У межах експерименту для віртуальної машини активовано DPDK-драйвер, який дає можливість перенести обробку пакетів з контексту ядра операційної системи до простору користувача, де виконуються високопродуктивні poll-mode цикли. Це дозволяє виключити ефекти блокування, характерні для систем на основі interrupts, та визначити "чисту" швидкодію механізму передачі пакетів через віртуальний комутатор. Як у випадку vHost-User, так і в реалізації vS2H, datapath-потік виконує обробку фреймів у контексті окремого виділеного ядра процесора, що забезпечує контрольовану та повторювану поведінку системи під навантаженням.

Особлива увага приділялася кількості операцій, які необхідно виконати при доставці одного пакета. Архітектура vHost-User передбачає, що основне навантаження з обробки дескрипторів покладається на PMD-потік Open vSwitch. У противагу цьому, в архітектурі vS2H значна частина цієї роботи переноситься у контекст QEMU-процесів, що дозволяє значно зменшити накладні витрати на перемикання контекстів. Унаслідок цього процесорний час використовується більш стабільно, а пропускна здатність не зменшується внаслідок короткочасних переривань чи конкурентного доступу до системних ресурсів. Саме ця відмінність стала ключовою для пояснення результатів.

Результати оцінювання пропускної здатності datapath наведено на рисунку 3.2. На рисунку показано, що при використанні великих пакетів, зокрема 1024 байтів, архітектура vS2H забезпечує приріст пропускної здатності до 14 % порівняно з традиційною реалізацією vHost-User. Середній приріст для всього діапазону розмірів пакетів становить близько 11 %, що свідчить про

систематичну перевагу vS2H у сценаріях, де вузьким місцем є обробка пакетів на рівні віртуального комутатора. Підвищення пропускної здатності пояснюється двома ключовими механізмами: відсутністю примусового переривання виконання потоку або процесу, який займається обробкою даних, та зменшенням навантаження на PMD-потік OVS, який виконує лише легкі lookup-операції, тоді як основну роботу виконує PD-процес у QEMU.

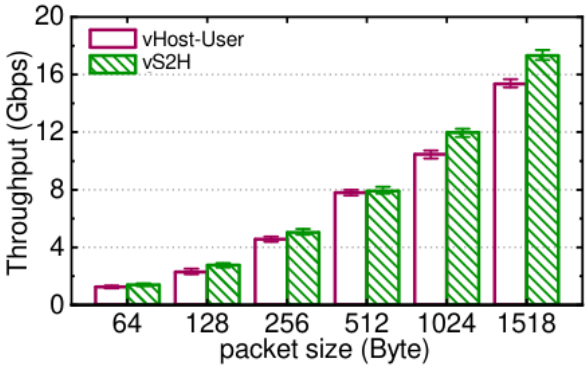



Рисунок 3.2 – Результати оцінювання пропускної здатності datapath

Оцінювання затримки datapath здійснювалося шляхом вимірювання часу проходження пакетів через віртуальний комутатор за умов постійного навантаження. На рисунку 3.3 показано, що архітектура vS2H демонструє збільшення затримки на 2–9 % у порівнянні з vHost-User. Попри те, що затримка зростає, її абсолютне значення залишається невеликим, досягаючи максимального збільшення приблизно на 2 мікросекунди. Такий результат пояснюється додатковою операцією копіювання дескрипторів пакетів, яка додається до шляху доставки в межах реалізації vS2H. Збільшення часу є сталим і не залежить від розміру пакета, що свідчить про те, що додаткові накладні витрати обумовлені алгоритмічними особливостями архітектури, а не масштабуються зі збільшенням обсягу переданих даних.

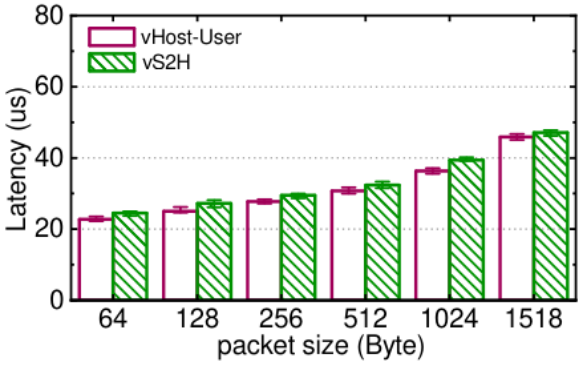



Рисунок 3.3 – Результати оцінювання затримки

Важливо підкреслити, що навіть із урахуванням невеликого зростання затримки загальна продуктивність системи в багатьох практичних сценаріях залишається кращою у vS2H завдяки значно вищій пропускній здатності. У системах, де підтримувані програми є чутливими до затримок, додаткові кілька мікросекунд зазвичай не впливають критичним чином на загальну продуктивність, особливо якщо мережевий канал працює під навантаженням, наближеним до максимального. Саме тому результати демонструють прийнятний і збалансований компроміс між пропускною здатністю та затримкою в рамках моделі vS2H.

Окремо слід відзначити, що вимірювання затримки та пропускної здатності у VNIO-дослідженнях мають особливу важливість, оскільки вони відображають властивості низькорівневої взаємодії між віртуальною машиною, віртуальним комутатором і фізичним середовищем передачі. На відміну від вимірювань на рівні TCP або прикладних протоколів, тут затримки вимірюються у мікросекундах і залежать не від алгоритмів congestion control або роботи сокетів, а від архітектурних рішень, що визначають розміщення та рух дескрипторів у пам’яті. Саме тому аналіз datapath є таким важливим при проєктуванні систем віртуалізованого мережевого введення-виведення.

Сукупність отриманих даних свідчить про те, що архітектура vS2H забезпечує стабільне й передбачуване покращення пропускної здатності в умовах високонавантаженого середовища. Показники, отримані в експериментальних вимірюваннях, демонструють, що механізм передачі даних у vS2H має меншу залежність від поведінки потоків ядра та процесів OVS, а

також менше страждає від ефектів втручання scheduler’a. Особливо це помітно при великих розмірах пакетів, коли частка накладних витрат на дескриптори в загальній вартості обробки зменшується, а переваги оптимізації PD-потоків стають найбільш вираженими. Водночас незначне збільшення затримки залишається стабільним і не перевищує рівнів, що мають практичне значення для типових NFV-сервісів. Таким чином, проведений аналіз дозволяє зробити висновок, що vS2H є ефективною архітектурою у контексті VNIO datapath. Вона забезпечує значне зростання пропускної здатності за відносно невеликих накладних витрат у вигляді збільшення затримки. Це створює підґрунтя для розуміння потенційних переваг vS2H у реальних системах віртуалізованих мереж, де продуктивність datapath має визначальне значення для роботи сервісів вищого рівня.

### 3.3 Оцінювання застосунків усередині віртуальних машин

Аналіз продуктивності каналу даних на рівні VNIO дає змогу оцінити базові можливості архітектури vS2H щодо обробки пакетів, однак у реальних умовах хмарних платформ та NFV-середовищ критичним є не лише throughput механізму доставки пакетів, а й те, як ці зміни позначаються на продуктивності застосунків, що працюють усередині віртуальних машин. Саме тому наступним етапом дослідження стало вимірювання швидкодії прикладних сервісів, розгорнутих у VM, за умов використання архітектур vS2H та традиційного vHost-User. Вимірювання на рівні застосунків дозволяють визначити, наскільки зміни у способі організації передачі даних впливають на ті компоненти системи, що безпосередньо взаємодіють з користувачами та мережевими потоками. Це суттєво, адже практична продуктивність мережевих платформ визначається не лише пропускною здатністю низькорівневих механізмів, а й тим, як ефективно працює мережева, реалізована в гостьових операційних системах.

Одним із перших застосунків, використаних для оцінювання, став високопродуктивний модуль IP lookup [25], реалізований у середовищі DPDK усередині віртуальної машини. Такий вибір не випадковий: IP lookup є базовою

операцією для багатьох мережевих функцій NFV, включаючи маршрутизатори, брандмауери та балансувальники навантаження. Його продуктивність чутливо реагує на зміну часу доставки пакетів, пропускної здатності каналу та накладних витрат на обробку дескрипторів. Використання DPDK-драйвера забезпечило мінімальний вплив ядра операційної системи на результати, дозволивши сфокусувати вимірювання саме на ефективності VNIO-механізму.

IP lookup у віртуалізованих і високопродуктивних мережевих системах є операцією, що забезпечує визначення маршруту подальшого передавання пакета на основі його IP-адреси призначення. Коли пакет надходить до мережевого пристрою або до віртуальної машини, яка виконує функції маршрутизації чи мережевої обробки, система повинна встановити, яким буде наступний вузол або інтерфейс для його пересилання. Для цього використовується таблиця маршрутизації, у якій зберігаються множини IP-префіксів і відповідні записи про напрямок руху трафіку. Пошук маршруту виконується за принципом найточнішої відповідності, тобто longest prefix match, коли з усіх префіксів, що частково збігаються з адресою призначення, обирається той, що має найбільшу довжину і, відповідно, є найбільш специфічним. Саме цей префікс визначає, куди має бути скерований пакет.

У сучасних NFV-платформах і системах, побудованих на DPDK, операція IP lookup виконується не у просторі ядра операційної системи, а в просторі користувача, що дозволяє уникнути затримок, спричинених обробкою переривань, переходом між рівнями привілеїв та традиційною маршрутизацією ядра. Замість цього пошук маршруту реалізується за допомогою високопродуктивних структур даних, оптимізованих для архітектури процесорів, які підтримують масивні послідовні обчислення та обробку великих потоків пакетів. Використання таких структур, як LPM-таблиці або спеціалізовані trie-дерева, забезпечує можливість виконувати десятки чи сотні мільйонів операцій пошуку за секунду, що робить IP lookup центральним компонентом програмних маршрутизаторів, віртуальних мережевих функцій та сервісних ланцюгів.

Особливе значення IP lookup має у контексті оцінювання ефективності архітектури мережевого введення-виведення віртуальних машин. Оскільки сама операція пошуку маршруту є відносно легкою з точки зору обчислювальних витрат, її продуктивність у віртуальних середовищах визначається переважно тим, наскільки швидко пакети доставляються у віртуальну машину та наскільки стабільною є ця доставка. У випадку інтенсивного трафіку затримки у VNIO-підсистемі накопичуються і впливають на кількість пакетів, які можуть бути оброблені протягом одиниці часу, що безпосередньо відображається на можливостях IP lookup. Якщо доставка пакетів від хоста до VM є нерівномірною або супроводжується примусовим переривання потоків планувальником операційної системи, то програма, яка виконує IP lookup, просто не отримує даних у стабільному темпі, і її пропускна здатність знижується, попри те, що власне обчислення маршруту може виконуватися дуже швидко.

У контексті дослідження IP lookup використовується як приклад високопродуктивного мережевого застосунку, чутливого до змін у конструкції VNIO. Він демонструє, яким чином архітектурні модифікації системи передачі пакетів можуть впливати на роботу реальних функцій, що складають основу віртуальних мережевих сервісів. Результати вимірювань продуктивності IP lookup служать індикатором того, наскільки ефективно працює фізичний і віртуальний шлях доставки пакетів, а також відображають взаємодію між користувацьким простором, механізмами DMA-передачі та внутрішньою структурою віртуального комутатора. Саме тому IP lookup є не просто окремою операцією, а репрезентативним тестовим навантаженням, яке дозволяє оцінити практичну користь від застосування моделі vSwitch-to-Hypervisor для покращення характеристик мережевої підсистеми віртуальних машин.

У дослідженні трафік для оцінювання роботи IP lookup генерувався спеціально так, щоб повністю завантажити VNIO-підсистему і при цьому не вносити додаткових накладних витрат на рівні стеку ядра чи транспортних протоколів. На відміну від TCP-тестів, де трафік формує сам qperf, для IP lookup використовувався TRex Traffic Generator, що працює поверх DPDK і дає

можливість генерувати точний, контрольований і високошвидкісний потік Ethernet-кадрів.

TRex надсилав до мережевого інтерфейсу хоста потоки пакетів фіксованих розмірів 64, 128, 256, 512, 1024 і 1518 байтів з максимально можливою інтенсивністю. Ці пакети не містили транспортних заголовків і не потребували обробки TCP або UDP, адже важливою була лише швидкість доставки кадрів до віртуальної машини, а не поведінка вищих рівнів протоколів. Саме тому трафік від TRex складався з «чистих» L2/L3-кадрів, які потрапляли безпосередньо до програми IP lookup у VM, створюючи умови для вимірювання максимально можливої продуктивності.

Після надходження у віртуальну машину пакети оброблялися DPDK-додатком, який виконував операцію визначення маршруту на основі LPM-таблиць. Оскільки DPDK працює у просторі користувача в режимі polling, IP lookup отримував пакети без затримок, пов’язаних із перериваннями, що дозволяло точно виміряти вплив саме VNIO-механізму на швидкість передачі та обробки трафіку.

Такий спосіб генерування трафіку забезпечував стабільне навантаження на віртуальний комутатор і на VNIO-шлях, при цьому уникаючи коливань, властивих ядру операційної системи або TCP-стеку. Завдяки цьому результати IP lookup у рисунку 3.4 відображають саме ефективність доставки пакетів до VM та їх подальшої обробки, а не поведінку додаткових протоколів чи мережевих механізмів. У результатах, наведених на рисунку 3.4, видно, що архітектура vS2H демонструє суттєве зростання пропускної здатності - до 20 % у випадку пакетів розміром 128 байтів і в середньому близько 9 % у межах усіх тестованих розмірів.

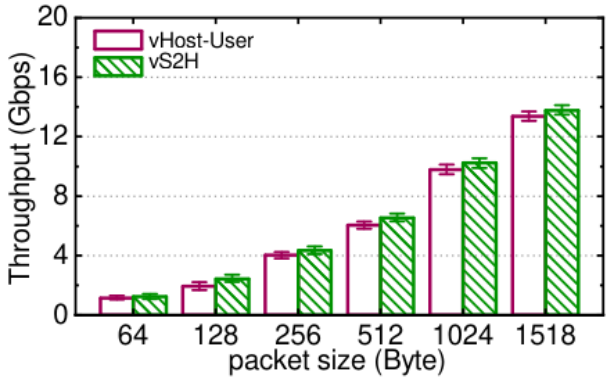



Рисунок 3.4 – Порівняння продуктивності для IP lookup

Зростання пропускної здатності пов’язане з тим, що гостьова система отримує пакети стабільнішими порціями та з меншою затримкою, що зменшує час простою між виконаннями lookup-операцій і покращує ефективність використання ядра процесора всередині VM.

Однак продуктивність IP lookup залежить не лише від пропускної здатності. На рисунку 3.5 продемонстровано, що затримка доставки пакетів у vS2H збільшується на 4–9 % у порівнянні з vHost-User.

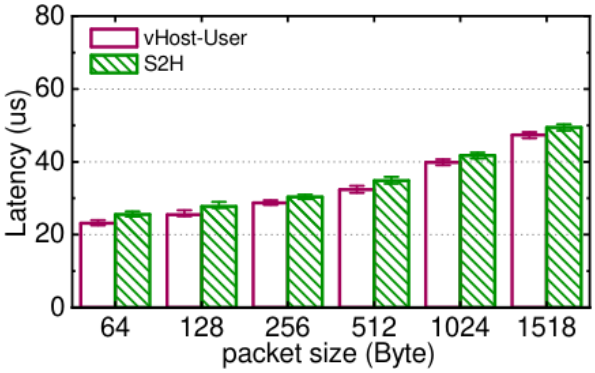



Рисунок 3.5 – Затримка доставки пакетів для IP lookup

Таке збільшення пояснюється додатковою операцією копіювання дескрипторів, властивою моделі vS2H. Попри додаткову затримку, загальна ефективність IP lookup залишається вищою у vS2H, оскільки швидше надходження пакетів у середньому компенсує невелике зростання затримок. Цей результат свідчить про те, що застосунки, виконані на основі DPDK, отримують більшу вигоду від збільшення пропускної здатності, ніж зазнають втрат від

невеликого росту затримки, що є важливим фактором для NFV-сервісів, орієнтованих на високу пропускну здатність.

Наступним критично важливим випадком для аналізу стала робота TCP-застосунків, які суттєво залежать від характеристик мережевого шляху, зокрема від стабільності затримки та поведінки стеку протоколів. Для вимірювання було застосовано утиліту qperf, яка дозволяє оцінити пропускну здатність TCP та затримку RTT при з'єднанні між віртуальною машиною та окремим фізичним сервером.

Трафік для аналізу роботи TCP-застосунків у дослідженні генерувався не штучним пакетоформувачем, а самим інструментом qperf, який створює реальний TCP-потік між двома вузлами. Це важливо, оскільки TCP-продуктивність не залежить від розміру Ethernet-пакета, а визначається поведінкою стеку протоколів, обробкою ACK-повідомлень, роботою вікна прийому та механізмами контролю перевантаження. Саме тому для TCP-аналізу використовуються не генератори на кшталт TRex, а засоби, які встановлюють повноцінне TCP-з'єднання.

Трафік формувався таким чином. На окремому фізичному сервері запускалася серверна частина qperf, яка перебувала у режимі очікування підключення. У віртуальній машині, що працювала під керуванням архітектури vS2H або vHost-User, запускалася клієнтська частина qperf з параметрами для вимірювання пропускної здатності та затримки TCP. Після встановлення з'єднання qperf починав надсилати потік TCP-повідомлень між VM і фізичним сервером. У режимі вимірювання пропускної здатності клієнт передавав безперервний потік даних у межах одного або кількох TCP-stream, а сервер приймав ці дані та фіксував кількість отриманих байтів за певний часовий інтервал. Таким чином визначалась ефективна TCP-пропускна здатність.

У режимі вимірювання затримки qperf формував послідовність коротких транзакцій типу «запит-відповідь», кожна з яких супроводжувалась часовою міткою. На основі цих міток обчислювався середній час проходження пакета в обидва боки (RTT). Оскільки трафік передавався через повний мережевий шлях, що включав VNIO-підсистему, віртуальний комутатор, фізичний NIC та

зовнішнє мережеве середовище, результати точно відображали вплив архітектурних рішень на TCP-поведінку реального застосунку. Таким чином трафік для аналізу TCP-застосунків генерувався безпосередньо механізмами qperf через встановлене TCP-з'єднання, що дозволило оцінити реалістичну пропускну здатність, затримку та стабільність роботи TCP у віртуалізованій мережевій інфраструктурі.

Результати, наведені на рисунку 3.6, демонструють помірне, але стабільне покращення пропускної здатності TCP при використанні архітектури vS2H - до 4 % у випадку пакетів розміром 128 байтів і близько 2 % у середньому.

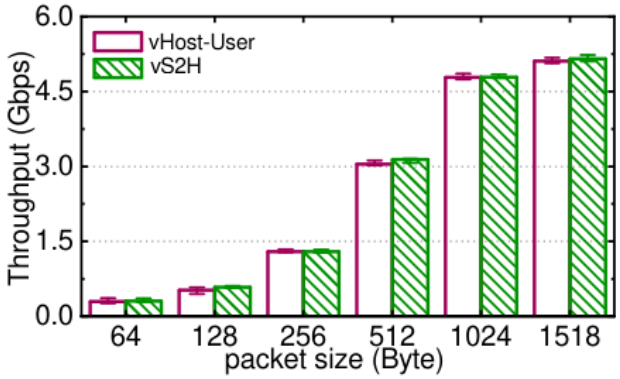



Рисунок 3.6 – Пропускна здатність TCP

На відміну від IP lookup, який безпосередньо залежить від швидкості доставки пакетів, TCP-продуктивність визначається також внутрішньою роботою стеку ядра Linux, механізмами контролю перевантаження та параметрами вікна прийому, тому вплив VNIO-архітектури є менш вираженим.

Що стосується затримок TCP, результати на рисунку 3.7 демонструють практично однакові величини для vS2H та vHost-User у всьому діапазоні розмірів пакетів.

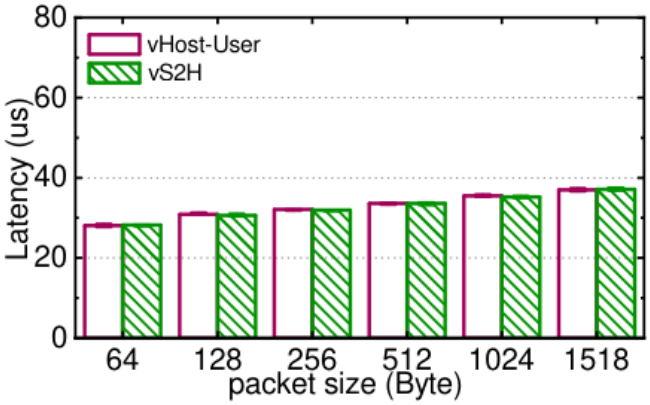



Рисунок 3.7 – Затримки TCP

Це пояснюється тим, що основну роль у формуванні TCP-затримок відіграють механізми, розташовані вище рівня VNIO, включаючи роботу мережевого стеку ядра гостьової ОС, механізми підтвердження пакетів та алгоритми керування чергами. Таким чином, зміни у VNIO-підсистемі не створюють істотного впливу на затримку TCP, що є очікуваним результатом і підтверджує коректність архітектурних модифікацій у vS2H: вони не порушують стабільності поведінки TCP.

Для оцінювання прикладних HTTP-навантажень було використано Nginx - один із найпопулярніших і найбільш оптимізованих вебсерверів. На відміну від низькорівневих мережевих інструментів, Nginx чутливо реагує на комбінацію факторів, включаючи дискові операції, роботу TCP, маршрутизацію та обробку HTTP-запитів. Тестування показало, що продуктивність Nginx практично не відрізняється між двома архітектурами: у vS2H сервер обслуговував 7895 HTTP-запити на секунду, а у vHost-User - 7231. Різниця не є статистично значущою у рамках подібних досліджень. Аналогічним чином затримка HTTP-відповіді залишалася дуже близькою: 107 мс для vS2H і 101 мс для vHost-User. Така поведінка пояснюється тим, що у Nginx основним вузьким місцем є стек протоколів усередині ядра та обробка HTTP-контенту, а не VNIO. Цей результат є важливою демонстрацією того, що vS2H не погіршує продуктивність реальних застосунків вебрівня, а отже, може бути використаний у широкому спектрі хмарних рішень без ризику негативного впливу на кінцевий користувацький досвід.

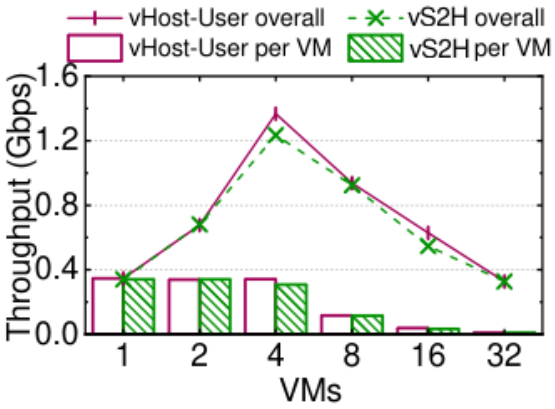
Таким чином, узагальнюючи результати тестування застосунків всередині віртуальних машин, можна стверджувати, що архітектура vS2H забезпечує помітні переваги у випадку високопродуктивних мережевих функцій, чутливих до швидкості доставки пакетів, зокрема DPDK-застосунків. Для TCP-орієнтованих програм поведінка vS2H практично ідентична до vHost-User, що є важливим для гарантування сумісності з наявними хмарними сервісами. У випадку комплексних прикладних рішень, таких як Nginx, продуктивність обох архітектур залишається близькою, а відмінності не перевищують нормальних статистичних коливань.

### 3.4 Масштабованість у багатокористувацьких сценаріях

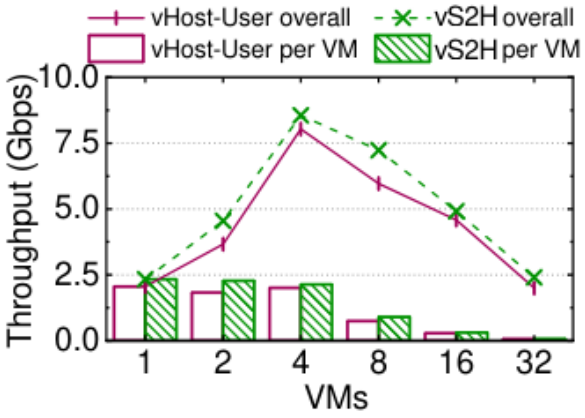
Оцінювання масштабованості системи в умовах віртуалізованого мережевого введення-виведення є критичним етапом аналізу архітектури vS2H, оскільки реальні хмарні платформи функціонують у середовищах із великою кількістю одночасно активних віртуальних машин, які конкурують за доступ до мережевих ресурсів. Продуктивність окремої VM має важливе значення, однак її поведінка в ізольованому експерименті не відображає повною мірою природи мережевих навантажень у багатокористувацьких системах, де ресурси розподіляються між десятками або навіть сотнями ізольованих середовищ. Саме тому дослідження масштабованості було зосереджено поведінці архітектури за умов збільшення кількості одночасно працюючих VM.

У багатокористувацькому сценарії важливо визначити, як поведінка архітектури змінюється при послідовному збільшенні числа VM від одиниць до десятків. Для цього було розгорнуто експеримент, у якому на фізичному сервері запускалося до тридцяти двох віртуальних машин, кожна з яких виконувала просту операцію пересилання трафіку. Усі VM працювали з однаковими параметрами пам’яті та CPU, а їхні мережеві інтерфейси були під’єднані до віртуального комутатора, що реалізовував VNIO-механізм відповідно до архітектури vS2H чи vHost-User. Два логічні ядра процесора були виділені для обробки потоків передачі даних у межах vS2H, що забезпечувало коректне

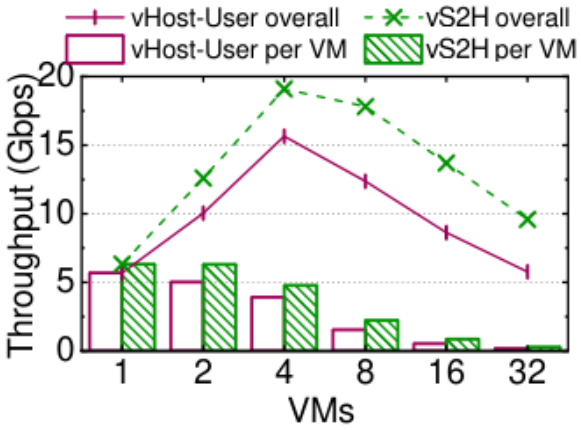
порівняння з вихідною системою і відтворення умов, максимально наближених до реальних. Для кожної конфігурації проводилися вимірювання сумарної пропускної здатності сервера, а також пропускної здатності, що припадає на одну VM. Отримані результати наведено на рисунку 3.8 для різних розмірів пакетів.




(а) пакети 64 байта




(б) пакети 512 байт




(в) пакети 1518 байт

Рисунок 3.8 – Порівняння пропускної здатності в багатокористувацькому сценарії

Результати демонструють, що обидві архітектури досягають свого максимуму пропускної здатності при запуску чотирьох віртуальних машин. Це пояснюється тим, що збільшення кількості VM дозволяє ефективніше завантажити PD-потоки та ресурс процесора, але після досягнення певного порогу починає проявлятися конкуренція за кеш-ресурси, пропускну здатність шини пам’яті та доступ до vSwitch. Незважаючи на спільні тренди, архітектура

vS2H демонструє помітно вищу продуктивність саме в зоні максимального навантаження, забезпечуючи приблизно на чверть більшу сумарну пропускну здатність у порівнянні з vHost-User. Це свідчить про те, що оптимізований механізм обробки дескрипторів у vS2H дозволяє краще балансувати між окремими потоками і ефективніше використовувати два виділені ядра процесора, що у підсумку веде до більш повного використання апаратного потенціалу системи. Подальше збільшення кількості віртуальних машин супроводжується поступовим зниженням пропускної здатності. Цей ефект спостерігається для обох архітектур і зумовлений не стільки особливостями VNIO-механізму, скільки фундаментальними обмеженнями апаратної платформи, яка не здатна забезпечувати безконфліктне обслуговування великої кількості одночасних потоків. Зменшення продуктивності при збільшенні кількості VM вище восьми є наслідком конкуренції за ресурси CPU, обмеженої пропускної здатності кешів і затримок, пов’язаних із синхронізацією доступу до структур даних у vSwitch. Утім важливо, що навіть за таких умов vS2H зберігає перевагу в сценаріях із середніми та великими розмірами пакетів. Для пакетів обсягом 1518 байтів перевага vS2H може досягати сорока відсотків у конфігурації з вісьмома віртуальними машинами, що свідчить про кращу масштабованість архітектури при високих навантаженнях на систему передачі даних. Для малих пакетів ситуація є більш чутливою. У вимірюваннях для 64-байтових кадрів vS2H демонструє продуктивність, близьку до vHost-User, інколи навіть трохи нижчу. Це пояснюється тим, що обробка надмірно великої кількості малих пакетів створює значне навантаження на механізм копіювання дескрипторів у vS2H, яке не компенсується перевагами стабільного виконання потоків. У випадках, де трафік складається переважно з мінімальних пакетів, частка накладних витрат зростає, і різниця між двома архітектурами вирівнюється. Однак такі навантаження є менш типовими для реальних хмарних сервісів, у яких значну частину трафіку становлять середні та великі IP-пакети. Загалом багатокористувацькі дослідження показали, що архітектура vS2H має кращу здатність до масштабування у складних багатокористувацьких середовищах.

Проведені експерименти демонструють, що архітектура vS2H добре масштабується як у багатокористувацьких умовах. Вона забезпечує високу пропускну здатність навіть тоді, коли ресурси обмежені й навантаження значне.

### 3.5 Висновки до розділу 3

В третьому розділі було проведено експериментальну оцінку продуктивності механізму vS2H, яка охоплює як низькорівневі характеристики VNIO-шляху, так і поведінку реальних мережевих застосунків у віртуалізованому середовищі. На основі єдиного стенда із сучасним стеком QEMU, Open vSwitch, DPDK і уніфікованих конфігурацій віртуальних машин сформовано методику, що дозволяє відокремити вплив архітектури vS2H від супутніх факторів і забезпечити відтворюваність результатів за критеріями RFC 2544. Використання TRex для генерації «чистого» L2-трафіку та qperf для TCP-вимірювань дало змогу дослідити як пропускну здатність, так і затримку на повному шляху від VM до фізичного сервера.

Отримані результати показали, що vS2H забезпечує відчутне зростання пропускної здатності VNIO datapath і високопродуктивних DPDK-застосунків (зокрема IP lookup), при цьому ціна у вигляді збільшення затримки є невеликою та практично не впливає на роботу TCP- і HTTP-сервісів, представлених Nginx. Окремо встановлено, що в багатокористувацьких сценаріях vS2H краще масштабується за кількістю одночасно активних віртуальних машин, особливо для середніх і великих розмірів пакетів, зберігаючи перевагу над vHost-User навіть за умов помітної конкуренції за апаратні ресурси. Сукупність цих спостережень підтверджує, що механізм vS2H є підходом, який дозволяє підвищити ефективність віртуалізованої мережевої підсистеми без суттєвих втрат для застосунків вищого рівня.

# РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1 Охорона праці

Метою роботи є підвищення рівня кібербезпеки віртуалізованих мережевих платформ шляхом удосконалення механізму ізоляції мережевого введення-виведення на основі архітектури vS2H та експериментальної оцінки його ефективності. Оскільки виконання експериментальної частини дослідження передбачає використання серверного обладнання, персональних комп’ютерів, мережевої апаратури та периферійних пристроїв, дотримання вимог охорони праці й техніки безпеки є необхідною умовою забезпечення надійності роботи технічних засобів і безпеки персоналу.

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [26]. У контексті роботи з комп’ютерними системами, серверним обладнанням, засобами віртуалізації та мережевими комутаторами дотримання вимог охорони праці дає змогу мінімізувати ризики ураження електричним струмом, негативного впливу мікрокліматичних факторів, шуму, вібрацій, електромагнітного випромінювання та загроз пожежної небезпеки.

Нормативними документами, що забезпечують охорону праці при роботі з ЕОМ є:

- НПАОП 0.00-7.15-18 “Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями” [27];
- Закон України “Про охорону праці”;
- Вимоги Правил пожежної безпеки в Україні (НАПБ А.01.001-2014) та Правил улаштування електроустановок (ПУЕ).

Дотримання цих документів забезпечує створення безпечного виробничого середовища і зменшує ймовірність виникнення техногенних інцидентів у процесі дослідження.

Експериментальна частина роботи виконувалася у спеціально підготовленому серверному приміщенні, умови якого відповідають установленим санітарно-гігієнічним, електротехнічним і пожежним нормам. Мікроклімат серверної повинен забезпечувати оптимальні умови для функціонування електронного обладнання та комфортну роботу персоналу. Температура у межах +22…24 °C, відносна вологість 40–60% та швидкість руху повітря близько 0,1 м/с підтримуються системами кондиціонування і припливно-витяжної вентиляції, які працюють безперервно і мають резервні засоби підтримання мікроклімату. Освітлення приміщення повинно забезпечувати не менше 300 лк на робочу поверхню, а використання світлих стін і стелі сприяє рівномірному розсіюванню світлового потоку та зменшенню втоми очей. Підлога виконується з лінолеуму або антистатичного покриття, що запобігає накопиченню електростатичного заряду. Розміщення та площа кімнати повинні забезпечувати достатній простір для встановлення обладнання та комфортні умови роботи персоналу. Для кожного робочого місця необхідно передбачати не менше 6,0 м² площі та 20,0 м³ об’єму приміщення [28]. Розміщення робочих або серверних приміщень у підвальних і цокольних поверхах не допускається. Рівні шуму та вібрації в приміщенні мають бути контрольованими й не створювати дискомфорту для працівників, при цьому серверна кімната не повинна безпосередньо межувати з приміщеннями, у яких можливі підвищені шумові або вібраційні навантаження. Електричне обладнання повинно підключатися через електророзетки із захисними контактами для приєднання нульового захисного провідника; використання двопровідних електричних мереж для живлення обладнання є неприпустимим. Пожежна безпека приміщення має забезпечуватися шляхом встановлення автоматичної системи пожежної сигналізації з димовими та тепловими сповіщувачами, а також оснащенням приміщення вуглекислотними вогнегасниками з розрахунку один вогнегасник на кожні 20 м² площі, але не менше двох одиниць на приміщення.

В даному розділі роботи було розглянуто основні нормативні документи та положення з охорони праці, які регулюють умови праці, використання комп’ютерних систем.

## 4.2 Підвищення стійкості роботи об’єктів господарської діяльності у воєнний час.

Для покращення стійкості роботи об’єктів вивчають фактори, які впливають на стійкість та оцінюють стійкість елементів і галузей виробництва проти уражаючих факторів ядерної, хімічної і біологічної зброї, стихійних лих і виробничих аварій. Щоб підвищити стійкість необхідно завчасно організувати і провести організаційні, інженерно-технічні й технологічні заходи [29].

Здійснення організаційних заходів передбачає завчасну підготовку всіх структур цивільного захисту, служб і формувань до надзвичайних ситуацій, в тому числі і військових дій. Вжиттям технологічних заходів підвищується стійкість роботи об'єктів шляхом змінювання технологічних процесів, режимів, можливих в умовах різних надзвичайних ситуацій. Інженерно-технічні заходи мають забезпечити підвищену стійкість виробничих споруд, технологічних ліній, устаткування, комунікацій об'єкта до впливу уражаючих факторів під час військових дій. При проведенні цих заходів необхідно враховувати конкретні умови об'єкта народного господарства. Проте є загальні організаційні інженерно-технічні заходи, які мають проводитись на всіх об'єктах.

Одним з найбільш важливих завдань в умовах воєнного часу і надзвичайних ситуацій є забезпечення захисту людей та їх життєдіяльності.

Для підвищення стійкості об’єктів господарювання та захисту людей необхідно:

– створити на об'єкті надійну систему оповіщення про загрози нападу противника, радіоактивне забруднення, хімічне і біологічне зараження, загрозу стихійного лиха і виробничої аварії;

– організувати розвідку і спостереження за радіоактивним забрудненням, хімічним і біологічним зараженням;

– організувати гідрометеорологічне спостереження за рівнем води, напрямком і швидкістю вітру, рухом і поширенням хмари радіоактивного забруднення, сильнодіючих отруйних речовин і отруйних речовин;

– створити фонд захисних споруд ЦО, запасів засобів індивідуального захисту і забезпечення своєчасної видачі їх населенню;

– завчасно підготуватись до масової санітарної обробки населення і знезаражування одягу;

– організувати взаємодію з установами охорони здоров'я для медичного обслуговування населення в умовах воєнного часу.

Також в умовах воєнного часу необхідно провести підготовку до евакуації населення, розміщеного в зонах можливих руйнувань і катастрофічного затоплення. Це передбачає завчасну підготовку місць евакуації, організацію прийому евакуйованого населення на територію населених пунктів. Окрім цього, необхідно забезпечити постачання продуктів харчування, питної води, предметів першої необхідності та провести заходи щодо морально-психологічної підготовки населення до виживання в умовах воєнного часу, забезпечити процес чіткого інформування про обстановку та правила дій і поведінки населення в надзвичайних ситуаціях воєнного часу [29].

Для забезпечення стійкості роботи об'єктів повинні проводитись інженерно-технічні заходи на мережах комунального господарства з метою захисту джерел тепла із заглибленням у ґрунт комунікацій. Котельні слід розміщувати в спеціальному окремо розміщеному приміщенні.

Якщо об'єкт одержує тепло з міської теплоцентралі, необхідно провести заходи для забезпечення стійкості трубопроводів і розподільних пристроїв, підведених до об'єкта. Теплова мережа має будуватися за кільцевою системою з прокладанням труб у спеціальних каналах зі з'єднанням паралельних ділянок. Для відключення пошкоджених ділянок мають бути встановлені запірно-регулюючі засувки, вентилі та ін. Ці пристосування необхідно розміщувати в оглядових колодязях, на території, що не завалюється при руйнуванні будівель.

Система каналізації має будуватись окремо: одна для дощових, друга для промислових і господарських вод. На об'єкті має бути не менше двох виводів з

підключенням до міських каналізаційних колекторів, а також виводи і колодязі з аварійними засувками на об'єктових колекторах з інтервалом 50 м на території, що не завалюється, для аварійного скидання неочищеної води в найближчі штучні та природні заглиблення.

На деяких промислових об'єктах є системи для забезпечення технології виробництва: для подання кисню, аміаку, стиснутого повітря та інших рідких і газових реактивів. Для цих систем розробляють заходи для попередження виникнення вторинних факторів зброї, стихійних лих та виробничих аварій і катастроф.

Створення резерву енергетичних потужностей за рахунок автономних пересувних електростанцій, а також місцевих джерел електроенергії. Підготовка автономних електростанцій до роботи за спеціальним режимом (графіком) для забезпечення технологічних процесів виробництва, для яких неможливі тривалі перерви в електропостачанні. З метою попередження аварій на електричних мережах необхідно установити автоматичну систему відключення при виникненні перенапруги. Повітряні лінії електропостачання замінити на підземно-кабельні. Створення необхідних запасів (резервів) паливно-мастильних матеріалів та інших видів палива й організація їх безпечного зберігання.

Щоб не допустити зупинки підприємства через дефіцит палива, необхідно підготуватись для роботи на різних видах палива: нафта, вугілля, газ. Для підвищення стійкості забезпечення водою слід провести такі заходи.

Необхідно створити основні і резервні джерела водопостачання. Як резервне джерело краще мати артезіанську свердловину, яку необхідно підключити до системи водопостачання. Крім того, воду можна брати з близько розміщеної природної водойми або спорудити штучну водойму чи резервуари з обладнанням пристроїв для збору і перекачування води. Всі ділянки водопостачання повинні бути заглиблені в ґрунт з обладнанням пожежних гідрантів і пристроїв для відключення пошкоджених ділянок. Локальні мережі водопостачання окремих великих підприємств варто з'єднати із 80 загальноміською системою водопостачання в єдине кільце.

Підвищенню стійкості забезпечення водою сприяє подавання води безпосередньо в мережу поза водонапірними баштами, спорудження обвідних ліній для подання води поза пошкодженими спорудами.

Завчасне вжиття заходів захисту джерел водопостачання, водопровідних споруд, свердловин і шахтних колодязів від забруднення радіоактивними речовинами, зараження хімічними і біологічними засобами. Підготовка меліоративних, гідротехнічних та іригаційних споруд і систем до експлуатації в надзвичайних умовах.

Для забезпечення виробництва продукції необхідні електроенергія, паливо, мастила, засоби захисту рослин, міндобрива, профілактичні й лікувальні препарати ветеринарної медицини, запасні частини, сировина та інші матеріально-технічні засоби. Забезпечення об'єктів цими ресурсами дасть можливість випускати необхідну продукцію в надзвичайних умовах мирного і воєнного часу. Тому повинні проводитись такі заходи, які б забезпечили стійкість постачання і сприяли підвищенню захисту мережі електро-, водо-, газопостачання, транспортних комунікацій і джерел постачання всім необхідним для забезпечення функціонування галузей сільського господарства в надзвичайних умовах.

З метою попередження аварій на електричних мережах необхідно встановити автоматичну систему відключення перенапруги. Повітряні лінії електропостачання слід замінити на підземно-кабельні. Газ використовується як паливо і на хімічних підприємствах у технологічному процесі. Для безперебійного забезпечення газом, газові мережі необхідно підводити до об'єкта з двох напрямків, які мають бути з'єднані в єдине кільце з обладнанням для можливого дистанційного автоматичного управління й у разі необхідності відключення пошкоджених ділянок. На великих підприємствах необхідно мати підземні ємності із закачаним резервним газом.

На підприємствах, де використовується пара, необхідно захистити джерела його постачання, заглибити в ґрунт комунікації паропостачання і встановити запірні пристосування.

Запас резервних матеріалів необхідно розраховувати на такі строки роботи підприємства, за які можливе відновлення регулярного постачання.

Передбачити, на випадок перебоїв в постачанні підприємствами-суміжниками, створення місцевих матеріалів, сировини для виготовлення комплектуючих виробів і інструментів силами свого підприємства [29].

Для підвищення стійкості та забезпечення збереження (відновлення) будівель і споруд в умовах воєнного часу необхідно:

– провести оцінку можливих ступенів руйнування будівель і споруд господарства населеного пункту, визначити обсяг невідкладних ремонтних робіт, потреби в будівельних матеріалах;

– створити і підготувати спеціальні формування для ремонтно-відновних, будівельних та інших робіт на об'єкті;

– розробити комплекс протипожежних заходів, які виключали б можливість виникнення масових пожеж.

Для забезпечення надійності системи управління і зв'язку потрібно організувати захищений пункт управління, забезпечити його засобами зв'язку, які б дали можливість швидко доводити сигнали ЦЗ до всіх виробничих підрозділів і населення у місцях проживання. При цьому необхідно здійснити планування збору даних про обстановку, передачу команд і розпоряджень в умовах впливу на об'єкт уражуючих факторів. Для підвищення стійкості системи управління і зв’язку в умовах воєнного часу необхідно організувати використання радіозасобів, засобів телефонного зв'язку  а також забезпечити зв’язок із колонами евакуйованого населення, що перебувають у дорозі, і відповідальними особами, які супроводжують їх під час евакуації, забезпечити дублювання ліній і каналів зв'язку.

# ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра було досліджено проблему забезпечення кібербезпеки у віртуалізованих мережевих платформах та розроблено удосконалений механізм ізоляції мережевого введення-виведення на основі моделі vS2H. Проведено теоретичне обґрунтування, моделювання архітектури та експериментальну оцінку ефективності запропонованого рішення, що дозволило підтвердити його переваги порівняно з існуючими підходами.

У першому розділі було проаналізовано архітектурні основи функціонування сучасних віртуалізованих мережевих платформ, включно з механізмами VNIO, особливостями роботи гіпервізора, віртуальних комутаторів і драйверів вводу-виводу. Встановлено, що традиційні моделі маршрутизації та обробки пакетів не повністю враховують вимоги кібербезпеки у багатокористувацьких хмарних середовищах. Особливу увагу приділено вразливостям, пов’язаним із порушенням ізоляції трафіку, перевантаженням vSwitch у високонавантажених сценаріях.

У другому розділі проведено аналіз проблем ізоляції у системах віртуалізованого мережевого I/O та критеріїв, що визначають їхній рівень захисту. Досліджено існуючі підходи до побудови VNIO-шляху та визначено архітектурні недоліки, які створюють умови для реалізації внутрішніх атак типу noisy-neighbor, міжвіртуальних витоків трафіку. На підставі аналізу сформульовано вимоги до підвищення ізоляції, узгодженої взаємодії з SDN-контролером та мінімізації впливу кіберзагроз на продуктивність.

У третьому розділі розроблено архітектуру та реалізовано прототип удосконаленого механізму vS2H, який передбачає перенесення частини функцій обробки пакетів із віртуального комутатора до гіпервізора. Сформовано методику експериментального оцінювання відповідно до рекомендацій RFC 2544 та виміряно пропускну здатність, затримки на стабільність VNIO. Експериментальні дослідження засвідчили, що запропонований механізм покращує ізоляцію мережевих ресурсів, знижує ризик міжвіртуальних атак та

забезпечує вищу передбачуваність продуктивності під навантаженням, зберігаючи або покращуючи пропускну здатність системи.

На основі проведених досліджень сформульовано такі узагальнені висновки:

- Запропонований механізм vS2H забезпечує підвищений рівень кібербезпеки за рахунок покращення ізоляції мережевих ресурсів та зменшення залежності від завантаженості віртуального комутатора. Це дозволяє ефективніше протидіяти внутрішнім DDoS-сценаріям і атакам noisy-neighbor.
- Архітектурні зміни в обробці пакетів дозволяють зменшити затримки та варіативність затримок, особливо у високонавантажених режимах, що важливо для сервісів реального часу та stateful-фільтрації.
- Методика експериментальної оцінки продуктивності, розроблена у роботі може бути використана для тестування альтернативних архітектур VNIO та механізмів безпеки у хмарних середовищах.
- Результати експериментів підтверджують, що механізм vS2H може бути застосований у практичних інфраструктурах, де необхідно одночасно забезпечити високу продуктивність і стійкість до внутрішніх загроз, зокрема у приватних хмарах, корпоративних дата-центрах і системах з підвищеними вимогами до безпеки.

Отже, поставлену мету дослідження досягнуто. Запропоновано та експериментально обґрунтовано механізм, що підвищує рівень кібербезпеки віртуалізованих мережевих платформ без критичного збільшення накладних витрат або погіршення їхніх експлуатаційних характеристик. Отримані результати можуть бути використані як основа для подальших досліджень.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Тимощук, В., & Тимощук, Д. (2022). Віртуалізація в центрах обробки даних-аспекти відмовостійкості. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 95-95.
2. Hypervisors: definition, types and solutions | Stackscale. (n.d.). Stackscale. https://www.stackscale.com/blog/hypervisors/
3. China, C. R., & Goodwin, M. (n.d.). Iaas, Paas, Saas: What's the difference? | IBM. IBM. https://www.ibm.com/think/topics/iaas-paas-saas
4. Xen vs. KVM - Comparison of Hypervisors | Storware BLOG. (n.d.). Storware. https://storware.eu/blog/xen-vs-kvm-comparison-of-hypervisors/
5. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56. https://doi.org/10.31891/2219-9365-2024-79-7
6. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220. https://doi.org/10.31891/2219-9365-2024-80-26
7. Тимощук, В., Долінський, А., & Тимощук, Д. (2024). ЗАСТОСУВАННЯ ГІПЕРВІЗОРІВ ПЕРШОГО ТИПУ ДЛЯ СТВОРЕННЯ ЗАХИЩЕНОЇ ІТ-ІНФРАСТРУКТУРИ. Матеріали конференцій МЦНД, (24.05. 2024; Запоріжжя, Україна), 145-146. https://doi.org/10.62731/mcnd-24.05.2024.001
8. Xen Project. (n.d.). Xen Project - Open Source Virtualization. https://xenproject.org/
9. Enabling Virtualization Features in Intel and AMD CPUs - SANS Setup and Troubleshooting General Guidance. (n.d.).

https://courseware.sans.org/workbook/LAB000-J01-WORKBOOK/troubleshooting/intel-amd-bios-vtx/

10. Open vSwitch. https://www.openvswitch.org/
11. Linux on IBM Systems. (n.d.). IBM. https://www.ibm.com/docs/en/linux-on-systems?topic=choices-using-linux-bridge
12. DPDK – The open source data plane development kit accelerating network performance. (n.d.). DPDK – The open source data plane development kit accelerating network performance. https://www.dpdk.org/
13. OpenFlow. (n.d.). F5, Inc. https://www.f5.com/glossary/openflow
14. Refined Speculative Execution Terminology. (n.d.). Intel. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/refined-speculative-execution-terminology.html
15. What Is a Side-Channel Attack? (n.d.). JumpCloud. https://jumpcloud.com/it-index/what-is-a-side-channel-attack
16. MITRE Corporation, “CVE-2018-1059.” https://www.cvedetails.com/cve/CVE-2018-1059/, 2018.
17. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.
18. Klots, Y., Titova, V., Petliak, N., Tymoshchuk, D., Zagorodna, N. Intelligent data monitoring anomaly detection system based on statistical and machine learning approaches. CEUR Workshop Proceedings, (2025), 4042, pp. 80 – 89
19. Perspectives, I. (2012, May 9). I/O Virtualization: Bringing It All Together. DataCenterKnowledge. https://www.datacenterknowledge.com/cloud/i-o-virtualization-bringing-it-all-together
20. Open vSwitch with DPDK Open vSwitch 3.6.90 documentation. https://docs.openvswitch.org/en/latest/intro/install/dpdk/
21. Donato, R. (2020, October 22). Virtual Networking Devices - TUN, TAP and VETH Pairs Explained. Packet Coders - Learn Network Automation.

https://www.packetcoders.io/virtual-networking-devices-tun-tap-and-veth-pairs-explained/

22. RFC 2544 (n.d.). IETF | Internet Engineering Task Force. https://www.ietf.org/rfc/rfc2544.txt
23. TRex Documentation. (n.d.). TRex. https://trex-tgn.cisco.com/trex/doc/index.html
24. How to schedule Quick Performance Overview (qperf) tool via cron. (n.d.). IBM. https://www.ibm.com/support/pages/how-schedule-quick-performance-overview-qperf-tool-cron
25. Efficient IP address retrieval using a novel octet based encoding technique for high speed lookup to improve network performance - Scientific Reports. (n.d.). Nature. https://www.nature.com/articles/s41598-024-84221-6
26. Mishko O., Matiuk D., Derkach M. (2024) Security of remote iot system management by integrating firewall configuration into tunneled traffic. Scientific Journal of TNTU (Tern.), vol 115, no 3, pp. 122–129.
27. Микитишин А. Г., Митник М. М., Стухляк П. Д. Телекомунікаційні системи та мережі. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2017. 384 с.
28. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник. Комп'ютерні мережі. Книга 1 [навчальний посібник] - Львів, "Магнолія 2006", 2013. - 256 с.
29. Закон України Про охорону праці Відомості Верховної Ради України (ВВР), № 49, 1992. 668 с.
30. Наказ міністерство соціальної політики України № 207 Про затвердження вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями від 14.02.2018.
31. ДСАНПІН 3.3.2.007-98: Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин №7 від 10.12.98.
32. Стручок В.С. Техноекологія та цивільна безпека. Частина «Цивільна безпека». Навчальний посібник. Тернопіль: ТНТУ. 2022. 150 с.

**Додаток А Публікація**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ІВАНА ПУЛЮЯ**

**М А Т Е Р І А Л И**

**XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ**

# «ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

**17-18 грудня 2025 року**

**ТЕРНОПІЛЬ**
**2025**

**УДК 004.056**
**Д. Мазурчак; В. Тимощук**
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

## ДОСЛІДЖЕННЯ МЕХАНІЗМУ ІЗОЛЯЦІЇ ПАМ'ЯТІ ДЛЯ ПІДВИЩЕННЯ БЕЗПЕКИ ВІРТУАЛІЗОВАНИХ МЕРЕЖЕВИХ СЕРЕДОВИЩАХ

**UDC 004.056**
**D. Mazurchak; V. Tymoshchuk**

## INVESTIGATION OF MEMORY ISOLATION TECHNIQUES FOR IMPROVING THE SECURITY OF VIRTUALIZED NETWORK SYSTEMS

У міру зростання масштабів хмарних дата-центрів і щільності віртуалізації саме порушення ізоляції пам'яті між віртуальними машинами та мережевими компонентами стає джерелом внутрішніх загроз, що важко виявляються класичними засобами мережевого моніторингу. Еволюція механізмів Virtualized Network I/O (VNIO) від повної віртуалізації до паравіртуалізованих рішень на основі virtio-net, vhost-net та vhost-user дозволила суттєво підвищити пропускну здатність VNIO-шляху, однак це відбулася ціною послаблення ізоляції між компонентами. У моделях VM-to-vSwitch (VM2vS) та vSwitch-to-VM (vS2VM) процеси віртуального комутатора (vSwitch) або самі віртуальні машини (VM) отримують прямий доступ до чужої пам'яті через механізми спільних буферів та відображення сторінок, що створює умови для DMA-атак, побічних каналів і порушення ізоляції орендарів у багатокористувацьких хмарних середовищах. Традиційні підходи до підсилення безпеки, зокрема використання vIOMMU або перенесення обробки даних у kernel space не забезпечують прийнятного балансу між безпекою та продуктивністю й часто призводять до помітного падіння пропускної здатності VNIO.

У роботі запропоновано архітектурну модель vSwitch-to-Hypervisor (vS2H). Ключова ідея полягає в перенесенні критичних операцій обробки дескрипторів пакетів із користувацького процесу vSwitch безпосередньо в гіпервізор. У межах моделі vS2H vSwitch позбавляється прямого доступу до пам'яті VM, а гостьові VM не працюють зі спільними I/O-буферами vSwitch. Єдиним посередником між пам'яттю орендарів і I/O-пам'яттю мережевих компонентів виступає гіпервізор, який уже є привілейованим і контрольованим елементом віртуалізаційної платформи. Такий підхід зберігає сумісність з інтерфейсом virtio, істотно зменшує поверхню атаки, локалізує можливі компрометації в межах одного рівня та мінімізує ризики витоку даних через спільні області пам'яті.

Ефективність моделі vS2H було експериментально оцінено на стенді на базі QEMU/KVM [1], Open vSwitch із підтримкою DPDK та генератора трафіку TRex. Результати вимірювань показали, що для каналу передачі даних запропонований механізм забезпечує середній приріст пропускної здатності близько 11% порівняно з класичним vhost-user. У сценаріях із розгортанням високопродуктивних DPDK-застосунків усередині VM спостерігається покращення стабільності обробки трафіку завдяки зменшенню конкуренції за CPU-ресурси між vSwitch та гостьовими VM. Отримані результати свідчать, що архітектура vS2H є перспективним напрямом розвитку безпечних VNIO-механізмів.

**Література**

1. Interactive cybersecurity training system based on simulation environments / D. Tymoshchuk et al. Measuring and computing devices in technological processes. 2024. No. 4. P. 215–220. URL: https://doi.org/10.31891/2219-9365-2024-80-26 (date of access: 30.11.2025).