НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«____» _____2021 р.

Дипломний проект

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»

на тему: Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж

Виконав (-ла): студент (-ка) 4 курсу, групи ІВ-72
(шифр групи)

Овчарова Юстіна Володимирівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач Алещенко О. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) проф. д.т.н. Сімоненко В.П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

«Комп'ютерні системи та мережі»

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Сергій СТІРЕНКО
«__» _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проект студента

Овчарової Юстіни Володимирівни

1. Тема проекту *Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж*

керівник проекту *Алещенко Олексій Вадимович, ст. викладач*

Затверджені наказом по університету від «11» травня 2021 року № 1139-с

2. Термін здачі студентом закінченого проекту (роботи) *2 червня 2021 р.*

3. Вихідні дані до проекту *технічна документація, теоретичні та статистичні дані, програмна реалізація конструювання топології та дослідження трафіку в мережі SDN за допомогою середовища емуляції Mininet та програмної мови Python.*

4. Зміст розрахунково-пояснювальної записки *опис предметної області, аналіз та дослідження методів для побудови та емуляції топології, програмна реалізація конструювання архітектури.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): *структурна схема топології на основі SDN-мережі, узагальнена схема алгоритму маршрутизації, блок-схема алгоритму маршрутизації.*

6. Консультанти проекту, з вказівкою розділів проекту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
нормоконтроль	д.т.н., проф. Сімоненко В. П.		

7. Дата видачі завдання 15.12.2020

Календарний план

№ п/п	Найменування етапів дипломного проекту	Терміни виконання етапів проекту	Примітки
1.	*Затвердження теми проекту*	*10.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-15.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2021-25.03.2021*	
4.	*Розробка структур окремих підсистем*	*25.03.2021-5.04.2021*	
5.	*Програмна реалізація системи*	*5.04.2021-15.04.2021*	
6.	*Оформлення пояснювальної записки*	*15.04.2021-20.05.2021*	
7.	*Захист програмного продукту*	*25.04.2021*	
8.	*Передзахист*	*23.05.2021*	
9.	*Захист*	*16.06.2021*	

Студентка _____ Овчарова Ю. В.

Керівник проекту _____ Алещенко О. В.

Анотація

В бакалаврській дипломній роботі реалізовано алгоритм побудови топології та маршрутизації пакетів OpenFlow для інтеграції SDN-мереж у концепцію сфери IoT.

Побудоване архітектурне рішення дозволяє обмін даними згідно з протоколом OpenFlow, що дозволяє централізоване управління, моніторинг і зручний збір статистики і спрощення обслуговування мережі.

Продукт був створений у середовищі емуляції комп’ютерних мереж Mininet, логіка програми реалізована за допомогою алгоритму передачі пакетів, розробленого на мові Python.

Annotation

In this work for a Bachelor's Degree the algorithm of construction of topology and routing of OpenFlow packages for integration of SDN-networks into the concept of IoT sphere is realized.

The built architectural solution allows data exchange according to the OpenFlow protocol, which allows centralized management, monitoring and convenient collection of statistics and simplification of network maintenance.

The product was created in the Mininet computer network emulation environment, the logic of the program is implemented using a packet transmission algorithm developed in Python.

Опис альбому

до дипломного проекту

на тему: «Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж»

Київ – 2021

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№ екземпляру	Примітки
	А4		Завдання на дипломний проєкт	2		
	А4	ІАЛЦ.467200.001 ВП	Відомість проєкту	2		
	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	3		
	А4	ІАЛЦ.467200.003 ПЗ	Пояснювальна записка	60		
	А4	ІАЛЦ.467200.006 Д3	Функціональна схема	1		
	А4	ІАЛЦ.467200.005 Д2	Структурна схема	1		
	А4	ІАЛЦ.467200.004 Д1	Принципова схема	1		

Зм	Арк.	№ докум	Підпис	Дата	**ІАЛЦ.467800.001 ОА**		
Розроб.		Овчарова Ю. В..			*Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж* *Опис альбому*	Літ.	Арк. 1 / Аркушів 1
Первір.		Алещенко О. В.					
Н.Контр.						КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72	
Затверд.							

Технічне завдання

до дипломного проекту

освітньо-кваліфікаційного рівня бакалавр

на тему: «Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж»

Київ – 2021

ЗМІСТ

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж.

Областю застосування є комп'ютерні мережі, використання у централізованій обробці даних та інтернет-безпеки.

.

2. ПІДСТАВИ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка топології системи та емуляція у призначеному середовищі для інтеграції SDN-мереж.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в мережі Інтернет з даних питань, довідники з програмування.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до розробленого продукту

- Зручність у використанні – розроблений продукт має бути зручним у користуванні;
- Актуальна інформація;
- Сучасний інтерфейс – розроблений продукт має мати графічний інтерфейс, реалізований сучасними програмними засобами.

5.2 Вимоги до програмного забезпечення

Для локальної версії проекту необхідне наступне програмне забезпечення:

- Операційна система Linux;
- Python 3 та вище;

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	15.12.2020
Аналіз завдання та огляд існуючих рішень	15.03.2021-5.04.2021
Розробка архітектури та загальної структури системи	5.04.2021-10.04.2021
Програмна реалізація системи	10.04.2021-1.05.2021
Доопрацювання і налагодження системи	1.05.2021-6.05.2021
Оформлення пояснювальної записки	2.05.2021-20.05.2021
Передзахист та проходження нормативного контролю	25.05.2021
Захист дипломного проєкту	16.06.2021

Пояснювальна записка
до дипломного проекту

на тему: «Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж»

Київ – 2021

ЗМІСТ

					ІАЛЦ.467800.003 ПЗ		
Зм.	***Арк.***	***№ докум.***	***Підпис***	***Дата***			
Розроб.		*Овчарова Ю. В.*			*Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж* *Пояснювальна записка*	***Лит.***	***Арк.*** / ***Аркушів***
Перевір.		*Алещенко О. В.*					*1* / *3*
Н. Контр.		*Сімоненко В. П.*				***КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72***	
Затверд.		*Стіренко С. Г.*					

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

IoT – (англ. Internet of things) Інтернет речей

SDN – (англ. software-defined network) програмно-конфігурована мережа

IDPS – (англ. intrusion detection and prevention systems) система виявлення та запобігання вторгнень

LAN – (англ. local area network) комп'ютерна мережа, яка покриває зазвичай відносно невелику територію або невелику групу будівель

ADSL – (англ. asymmetric digital subscriber line) технологія широкосмугового доступу, яка забезпечує передачу швидкісного цифрового сигналу звичайною аналоговою телефонною лінією, та дозволяє одночасно користуватися телефоном і Інтернетом

QoS – (англ. quality of service) набір методів для управління ресурсами пакетних мереж

CDPI – (англ. control data plane interface) драйвер інтерфейсу управління площиною даних

NBI – (англ. nourthbound interface) програмний інтерфейс, за допомогою якого додаток представляє низькорівневі деталі в архітектурі системи з додатком

SBI – (англ. southbound interface) програмний інтерфейс, за допомогою якого додаток звертається до нижчому в архітектурі системи, з додатком

OVSDB – (англ. Open vSwitch Database) мережево доступна система баз даних

ONOS – (англ. Open Network Operating System) контролер SDN з відкритим кодом для створення рішень SDN/NFV наступного покоління

API – (англ. Application Programming Interface) опис способів (набір класів, процедур, функцій, структур), якими одна комп'ютерна програма може взаємодіяти з іншою програмою.

ВСТУП

За останні кілька років Інтернету речей (IoT) став однією з найважливіших технологій 21 століття. Тепер, коли ми можемо підключити повсякденні предмети - кухонні прилади, машини, термостати, няні - до Інтернету через вбудовані пристрої, стало можливим безперервне спілкування між людьми, процесами та речами.

За допомогою недорогих обчислень, хмари, великих даних, аналітики та мобільних технологій фізичні речі можуть обмінюватися та збирати дані з мінімальним втручанням людини. У цьому гіперпов’язаному світі цифрові системи можуть записувати, контролювати та регулювати кожну взаємодію між пов’язаними речами. Фізичний світ відповідає цифровому - і вони співпрацюють.

Збільшилось використання у сфері IoT, який останнім часом стає все більшою сферою інтересів, оскільки широко використовується для численних додатків та пристроїв, таких як бездротові датчики, медичні пристрої, чутливі домашні датчики та інші пов'язані пристрої IoT. Через попит на швидкий випуск нових продуктів IoT на ринок, аспекти безпеки часто залишаються поза увагою, оскільки для вивчення всіх можливих вразливостей потрібен час. Оскільки пристрої IoT засновані на Інтернеті та містять конфіденційну інформацію, було порушено питання безпеки, і кілька дослідників вивчають методи підвищення рівня безпеки серед цих типів пристроїв.

З урахуванням постійно зростаючого Інтернету та взаємозв’язку серед усього, потреба в безпеці є вимогою часу. Інтернет речей (IoT) [1], який пов'язує кожен об'єкт або пристрій за допомогою мережевих можливостей, є областю, що викликає велике занепокоєння, пов'язану з безпекою. Об'єкти включають датчики домашньої автоматизації, медичне обладнання, автомобільні датчики, ядерні реактори та будь-які критичні для життя датчики реального часу [2]. Це означає, що відсутність безпеки у сфері Інтернету речей може становити небезпеку для життя людей.

IoT складається з багатьох різнорідних пристроїв, які використовують різні протоколи. Кожен протокол дотримується різних механізмів доступу та заходів безпеки. Але єдиного механізму безпеки все ще немає в IoT.

Звичайні підходи до безпеки, такі як системи виявлення та запобігання вторгненню (IDPS), брандмауер, розгортаються на мережевих пристроях, що що мають безпосереднє або пряме з'єднання з Інтернетом або зовнішньою непропрієтарною мережею, для захисту від зовнішніх атак. Але у випадку IoT, який є бездоганним та безмежним, контроль доступу до мережі стає складнішим.

Програмно-конфігурована мережа (SDN) [3], яка є новою парадигмою інтелектуальних мереж, забезпечує можливості вирішення питань, пов'язаних з IoT. Застосовуючи конфігурацію та управління мережею SDN, це можна значно спростити. Широке визнання галузей SDN свідчить про те, що SDN може встановити більш тісний зв'язок в екосистемі IoT.

В цій роботі описується необхідність у використанні SDN та її розвитку, конструювання архітектури IoT на основі SDN та запропонована система безпеки на основі архітектури SDN-IoT.

РОЗДІЛ 1

АРХІТЕКТУРА СИСТЕМИ НА ОСНОВІ SDN ДЛЯ IoT

1.1. Огляд системи розумного будинку, заснованого на концепції Інтернету речей (IoT)

Розумний дім був описаний як сучасне застосування повсюдних обчислень, що включає інтелект у управління житлом та експлуатацію для комфорту, охорони здоров'я, безпеки, безпеки та енергозбереження. Ця точка зору, також прийнята в цьому оглядовому документі, ілюструє збіжність двох взаємодоповнюючих перспектив функціонування розумного будинку: орієнтованої на користувача та систему, що, відповідно, концентрується на комфорті мешканців та ефективності будівельної системи. Підхід до розумних будинків, орієнтований на користувача чи додому, розпочався в першій половині 20 століття і розвивався десятиліттями. Погляд, орієнтований на систему чи будівлю, з'явився лише з розвитком інформаційно-комунікаційних технологій та появою інтелектуальної енергетичної інфраструктури.

IoT робить перехід від функціональності до підключення та прийняття рішень на основі даних, що означає, що пристрій може стати більш корисним, якщо він взаємопов'язаний з іншими пристроями. Однак IoT - це не просто набір пристроїв та датчиків, підключених один до одного в дротовій або бездротовій мережі - це щільна інтеграція віртуального та реального світу, де відбувається спілкування між людьми та пристроями. Його можна вважати переплетеним середовищем мереж різного розміру, що становить велику глобальну мережу.

Розумний дім - це органічна комбінація різних підсистем, пов'язаних із побутом через передові технології, такі як волоконно-оптичний композитний кабельний дім [4]. Він може обмінюватися ресурсами та спілкуватися вдома, а ми можемо обмінюватися інформацією із вашою домашньою зовнішньою мережею через ваш домашній розумний шлюз. Його головна мета - забезпечити людей ефективним, комфортним, безпечним, зручним та екологічно чистим середовищем життя, що інтегрує систему, сервіс та управління.

Розумний дім – це використання комп’ютерних технологій, технологія управління, технологія відображення зображень та комунікаційні технології будуть пов’язані через мережу різних об’єктів разом для задоволення вимог до автоматизації всієї системи для забезпечення більш зручного управління та менеджменту [5]. Традиційна реалізація розумного будинку, як правило, контролює та передає споруди за допомогою дротових проводів, важко позбутися обмежень різних кабелів, вартість встановлення висока, а масштабованість системи також погана. Система розумного будинку, заснована на бездротовій технології сенсорних мереж, дозволяє не тільки позбутися кайданів кабелів, зменшити вартість монтажу, але й значно покращити масштабованість системи.

Нижче наведено кілька основних функцій розумного будинку [4]:

- Розумний дім може реалізувати взаємодію між користувачем та електромережевим підприємством, отримати інформацію про споживання електроенергії та ціну на електроенергію, встановити план споживання електроенергії тощо, керувати науковим та раціональним використанням електроенергії та захищати свідомість сім'ї енергозбереження та охорони навколишнього середовища;
- Розумний дім може підвищити комфорт, безпеку, зручність та інтерактивність домашнього життя та оптимізувати стиль життя людей;
- Розумний будинок може підтримувати віддалену оплату;
- Розумний дім може контролювати та взаємодіяти з будинком за допомогою телефону, мобільного телефону та віддаленої мережі, виявляти ненормальну та своєчасну обробку;
- Розумний дім реалізує в режимі реального часу показ показань лічильника та службу безпеки лічильника води, лічильника електричної енергії та лічильника газу, які забезпечують більш

зручні умови для високоякісного обслуговування;

- Підтримка бізнесу "потрійних мереж" та ідеального інтелектуального сервісу.

Побудувавши внутрішню комунікаційну мережу, ми реалізуємо домашню систему кондиціонування та інтелектуальну побутову мережу шляхом взаємозв’язку оптоволоконних мереж. Завдяки інтелектуальним інтерактивним терміналам, розумним розеткам, розумним електроприладам тощо ми досягаємо побутової техніки, яка автоматично збирає інформацію про електроенергію, аналіз, управління; а побутова техніка досягає економічної роботи та контролю енергії [6, 7]. За допомогою телефону, стільникового телефону, Інтернету та інших засобів система може дистанційно керувати будинком та іншими послугами. Завдяки інтелектуальним інтерактивним терміналам ми також забезпечуємо виявлення диму, виявлення витоку газу, протиугінні засоби, допомогу в надзвичайних ситуаціях та інші функції домашньої безпеки, а також здійснюємо автоматичний збір та управління інформацією про лічильники води, лічильники газу та майстер осередку центру підтримки та управління майном мережі, а також отримати інформацію про домашню безпеку, дозволену односторонню передачу та інші послуги.

Розумну домашню систему зв'язку можна розділити на зовнішню мережу, шлюз та внутрішню мережу на 3 частини. Зовнішня мережа може бути стільниковою локальною мережею, мережами кабельного телебачення, телефонними мережами та Інтернетом, переважно з використанням більш зрілих технологій. Інтранет використовується для взаємозв’язку різних побутових приладів у межах сім’ї, обладнання, локальної мережі, через величезну різноманітність підключених пристроїв мережа також продемонструвала велику різноманітність форм. Домашні мережі за своїми функціями в основному поділяються на три категорії: мережа управління для управління функціями, мережа передачі даних для обміну повідомленнями даних та мультимедійна мережа для передачі аудіо та відео. Домашній шлюз –

це мережевий приєднувальний пристрій, який з'єднує домашню інтрамережу та екстрамережу та здійснює доступ до інтрамережі до екстрамережі, щоб надати їй функцію управління взаємозв'язку пристроїв у домі. У той же час домашній шлюз дозволяє дому застосовувати різні мережеві технології та використовувати шлюзи, що забезпечують можливості з'єднання для різних підмереж зв'язку, щоб мережеві пристрої в кожній підмережі могли взаємодіяти між собою.

- Мережа побутової техніки: побутова техніка (холодильники, кондиціонери, телевізори, мікрохвильові печі, пральні машини, освітлення тощо) складає мережі за допомогою дротових або бездротових з'єднань для обміну інформацією;
- Безпека: включаючи захист навколишньої території, домашній відеодомофон, контроль доступу, охоронну сигналізацію, пожежу, витоки газу, розливи води тощо;
- Високошвидкісний доступ до інформації: Інтернет, відеотелефони, стільниковий LAN-доступ до дому через шлюз;
- Житлові послуги: Центр управління громадою може контролювати обладнання та навколишнє середовище, що перебувають під його юрисдикцією, та керувати ними.

Основним фактором розумної домашньої системи є домашня внутрішня мережа зв'язку, яка в основному включає дві частини: розумний домашній шлюз та домашній розумний вузол датчика. Smart Home Gateway - це сімейний центр управління та налаштування ресурсів для завершення домашньої мережі та управління вузлами та інших функцій. Шлюз розумного будинку за допомогою мережевої технології з'єднує кожен вузол комутатора датчика в домашній мережі, реалізує управління та контроль внутрішньої мережі розумного будинку за допомогою стандартного протоколу зв'язку та служить інтерактивним інтерфейсом інформації домашньої та зовнішньої мережі.

Система розумного будинку – це свого роду система управління, заснована на одночіповому комп'ютері, який може отримати доступ до домашніх пристроїв та керувати ними за допомогою телефону та Інтернету. Розумна система управління домашньою мережею за допомогою збору даних, модуля керування командами та модуля протоколу TCP/IP для досягнення мережі моніторингу безпеки, управління командами поділяється на три режими:

- телефонне дистанційне управління;
- мережеве дистанційне управління;
- місцеве управління.

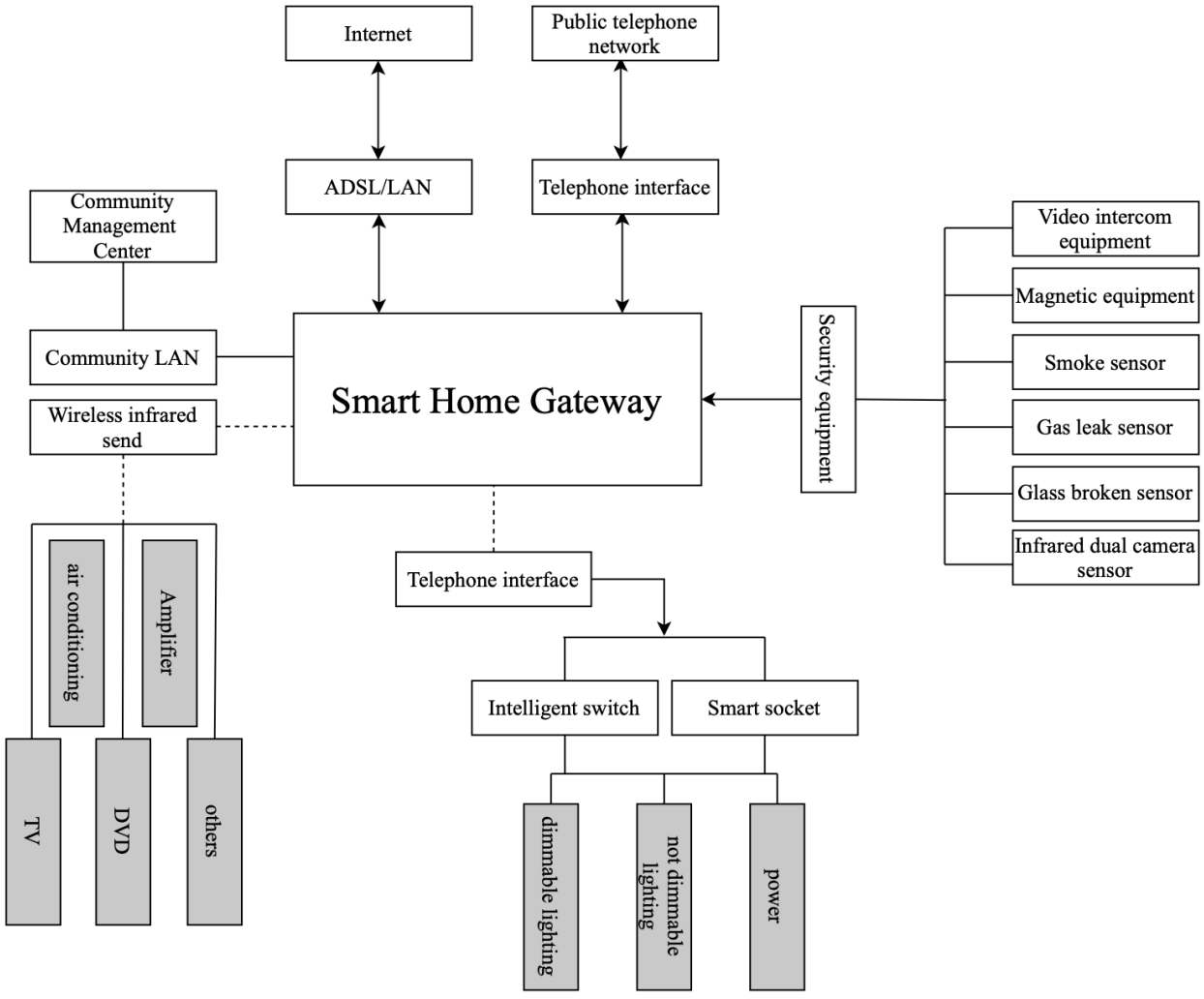


Рис. 1.1 – Мережева структура розумного будинку

1.2. Огляд програмно-конфігурованої мережі (SDN)

1.2.1. Базові поняття та принципи мережі

Програмно-конфігурована мережа (SDN) є важливою зміною парадигми комп'ютерних мереж за останні 10 років. Концепція SDN настільки потужна, що потенціал її застосування може бути легко сприйнятий поза початковим випадком використання великих мереж центрів обробки даних. Це сприйняття спонукає дослідити потенційне використання SDN в контексті домашніх мереж, хоча домашні середовища не були рушійним сценарієм SDN у перші роки його розвитку.

Вся ідея SDN полягає в тому, щоб запровадити розділення між фактичними функціями переадресації мережі, яка може бути виконана в апаратному забезпеченні (колективно названо як пересилання мережі) та керування цими функціями, які можуть бути виражені в програмному забезпеченні (колективно названі як контрольна площина мережі). Це відокремлення проблем зберігає обіцянку більш гнучкої мережевої архітектури, яка дозволяє інновації та полегшити завдання управління мережею.

У програмно-конфігурованій мережі інженер мережі або адміністратор може сформувати трафік з централізованої консолі керування без необхідності торкнутися окремих вимикачів у мережі. Централізований контролер SDN спрямував перемикачі, щоб доставити мережеві послуги, де вони не потрібні, незалежно від конкретних зв'язків між сервером та пристроями.

Даний процес – це відхід від традиційної мережевої архітектури, в якій окремі мережеві пристрої роблять рішення щодо трафіку на основі їх конфігурованих таблиць маршрутизації. SDN мала роль у мережі протягом десятиліття зараз, і її еволюція та роль продовжували розвиватися.

1.2.2. Архітектурні компоненти

Основна архітектура SDN використовує абстракції на основі модульності, подібні до формальних методів програмної інженерії. Типова архітектура SDN розділяє такі процеси, як конфігурація, розподіл ресурсів, пріоритезація трафіку та переадресація трафіку в базовому обладнанні, використовуючи трирівневу

структуру, що складається із програми, управління та площини даних, як зазначено нижче.

- Рівень 1 – Площина даних (переадресація): Площина даних - це сукупність фізичних компонентів мережі (комутатори, маршрутизатори, віртуальне мережеве обладнання, брандмауери, прилади середньої скриньки тощо), майже такі ж, як у звичайних мережах. Він спрямований на ефективну переадресацію мережевого трафіку на основі певного набору певних правил, заданих Рівнем 2 - Площина управління. Таким чином, технологія SDN робить апаратну (фізичну) інфраструктуру мережі досить гнучкою, видаляючи інтелект та ізольовану конфігурацію для кожного елемента мережі, одночасно переміщуючи ці функціональні можливості в площину управління.
- Рівень 2 – Площина управління: Площина управління - це логіка, яка визначає, як можна перенаправляти трафік через мережу з одного вузла на інший на основі вимог кінцевого користувача. Така логіка управління вбудована у зовнішні елементи програмного забезпечення, які називаються контролерами SDN. Контролер SDN обробляє всі пристрої переадресації рівня 1, перекладаючи індивідуальні вимоги до програм та бізнес-цілі (наприклад, встановлення пріоритетів трафіку, контроль доступу, управління пропускною здатністю, QoS тощо) у відповідні програмовані правила та оголошення їх на площину даних. Впроваджуючи програмованість за допомогою цих правил, таблицями потоків можна маніпулювати окремими елементами та в режимі реального часу на основі продуктивності мережі та вимог до послуг.
- Рівень 3 – Площина додатків: Площина додатків - це рівень, який включає всі програми та послуги мережі. Концептуально площина додатків розташована над площиною управління, щоб забезпечити можливість додатків, що спілкуються з площиною даних, через запит мережевих функцій із площини управління, виконуючи завдання, пов'язані з мережею. Площина додатків використовує API для

захоплення окремих параметрів програми (затримка, пропускна здатність, затримка тощо), на основі яких контролер SDN конфігурує окремі елементи мережі в площині даних для ефективного пересилання трафіку [8, 9].

Розглянемо архітектурні компоненти та їх взаємодію наглядно на рис. 1.2 на прикладі OpenFlow – протоколу управління процесом обробки даних, переданих через мережу передачі даних маршрутизаторами та комутаторами, що реалізує технологію програмно-конфігурованої мережі.

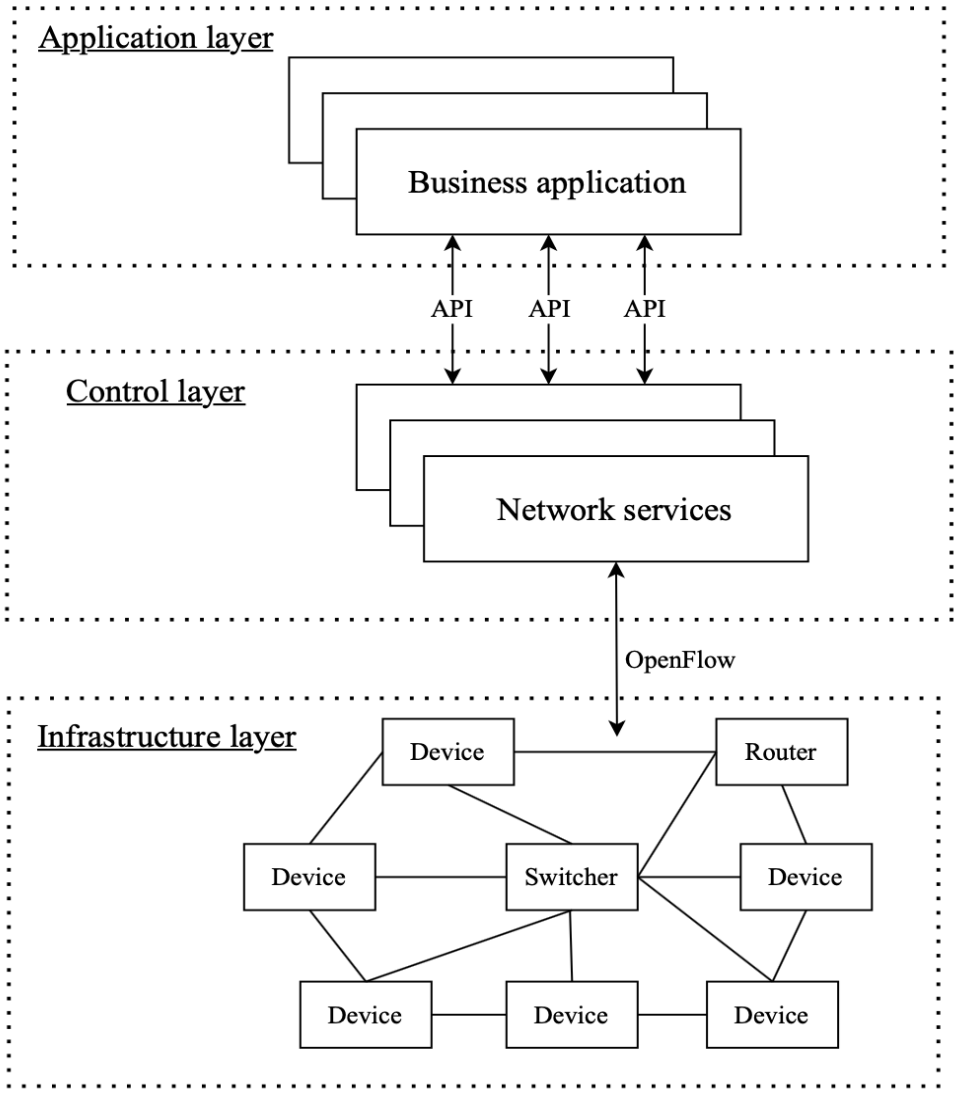


Рис. 1.2 – Ілюстрація елементів стандарту OpenFlow SDN

Стандарт OpenFlow включає три основні елементи, а саме комутатори, контролери та протоколи. Комутатори OpenFlow відповідають за пересилання

пакетів даних (тобто площину даних) відповідно до правил, створених і підтримуваних контролером SDN. У OpenFlow контролер SDN є основним компонентом для підтримки мережевих програм шляхом визначення правил, які слід зберігати та застосовувати комутаторами.

1.2.3. Протоколи та стандарти зв’язку

У цьому розділі представлені найвизначніші стандарти зв'язку SDN, так званий інтерфейс прикладного програмування (API), а також наведені приклади нещодавніх спроб вивчити розвиток цих стандартів у напрямку до системи 5G.

Стандарти зв’язку «півдня»

Стандарт OpenFlow: Стандарт OpenFlow включає три основні елементи, а саме комутатори, контролери та протоколи (API, що стосується півдня), як показано на малюнку 2-2. Перемикачі OpenFlow відповідають за переадресацію пакетів даних (тобто площину даних) відповідно до правил, створених і підтримуваних контролером SDN [10]. У OpenFlow контролер SDN є основним компонентом для підтримки мережевих додатків шляхом визначення правил, які слід зберігати та застосовувати комутаторами. Доступні кілька контролерів OpenFlow, такі як OpenDayLight, Floodlight, ONOS, Ryu, POX та NOX. Контролер SDN на основі OpenFlow може використовувати два типи інтерфейсів для створення правил на комутаторах, а саме: протокол OpenFlow і протокол управління Open vSwitch Database (OVSDB).

- Протокол OpenFlow: протокол OpenFlow (ONF), який підтримується та оновлюється ONF, є першим і найбільш відомим API зв'язку на південь. Цей протокол використовує захищений та зашифрований канал (TCP / TLS) для здійснення управління комутаторами. Він включає набір повідомлень для різних ситуацій: встановлення та конфігурація каналу управління (наприклад, Hello, Echo Request / Reply, Features Request / Reply, Set-Config), отримання (Packet-in) та перенаправлення (Packet-out) пакетів даних та управління правилами OpenFlow (Flow-mod). Протокол OpenFlow може координувати всі комутатори з підтримкою OpenFlow.

- Протокол управління OVSDB: визначений у RFC 7047 [11], цей протокол використовує набір операцій, доступних у Open vSwitch (програмне розширення) для управління правилами OpenFlow. Такі операції дозволяють вставляти, оновлювати та видаляти правила пересилання безпосередньо в базу даних Open vSwitch. Як наслідок, протокол управління OVSDB обмежений для використання у віртуальних комутаторах на основі Open vSwitch.

Як приклад використання вищезазначених двох протоколів, заснованих на OpenFlow, дослідження, проведене в роботі [12], розгортає сервісну функцію декількох орендарів до ланцюгових функцій мережі, використовуючи OpenStack. Він робить це, використовуючи протокол управління OVSDB через Neutron Open vSwitch Agent для забезпечення зв'язку з VNF, а протокол OpenFlow через контролер POX для контролю пропускної здатності правил OpenFlow.

Стандарт пересилання та розділення елементів управління (ForCES): ForCES - це стандарт SDN, визначений робочою групою Інженерного проектування (IETF) [13]. У ForCES поділ площини відбувається шляхом розділення мережевих елементів на дві сутності: переадресовувальні елементи та елементи управління. Перші представляють площину даних і містять як фізичні, так і віртуальні комутатори. ForCES моделює FE, визначаючи один або кілька класів логічних функціональних блоків, реалізованих за допомогою моделювання на основі XML. LFB містить вхідні та вихідні порти і діє як ресурс обробки пакетів, виконуючи різні функції, такі як фільтрація, класифікація та вимірювання. Кілька екземплярів LFB в одному і тому ж FE можуть бути з'єднані в спрямованому графіку для створення мережі. Кожен LFB забезпечує робочі параметри, можливості та події CE, який діє як контролер SDN. CE використовує протокол ForCES як API на південь для здійснення контролю за LFB.

Як приклад, у дослідженні [14] пропонується архітектура NFV / SDN із використанням стандарту ForCES, де інфраструктура NFV (NFVI) включає

гіпервізор LFB, що дозволяє створювати FE / LFB як VNF та CE як сутності EMS. Крім того, NFVI забезпечує LFB, які діють як віртуальні комутатори для взаємозв'язку цих VNF. Крім того, [15] пропонує підхід ForCES для оцінки архітектури PoC при застосуванні для віртуалізації компонентів 4G Evolved Packet Core (EPC).

Стандарт розширеного обміну повідомленнями та протоколу присутності (XMPP): XMPP - це загальний стандарт зв'язку, що пропонує обмін повідомленнями та обмін інформацією між клієнтами через централізовані сервери [16]. Подібно до протоколу передачі повідомлень (SMTP), кожного клієнта XMPP можна ідентифікувати за допомогою ідентифікатора, який може бути таким самим простим, як адреса електронної пошти. Клієнтські машини встановлюють зв’язки з центральним сервером, щоб повідомляти про їх присутність, який підтримує адреси контактів і може повідомляти інші контакти про те, що певний клієнт перебуває в мережі. Клієнти спілкуються між собою за допомогою чатових повідомлень, які передаються на відміну від опитування, що використовується в електронних листах SMTP / POP. Кожен об’єкт SDN, такий як віртуальна машина, комутатор та гіпервізор, може мати клієнтський модуль XMPP, який очікує інструкцій від сервера XMPP щодо автентифікації та переадресації трафіку, тоді як кожен клієнт оновлює свою конфігурацію відповідно до запиту сервера. XMPP API - це стандартизація IETF протоколу Jabber і конфігурована для використання з TCP-з’єднаннями в RFC 6121. Ряд реалізацій XMPP з відкритим кодом також доступні з варіаціями, що використовуються в програмах, зокрема Google, Skype, Facebook та багатьох іграх.

Стандарт Cisco OpFlex: Стандарт Cisco OpFlex [17] набрав обертів серед південних API, оскільки він полегшує зв'язок площини управління та передачі даних, реалізуючи мову програмування у фізичному та віртуальному середовищах. У порівнянні зі стандартом OpenFlow, який централізує функції управління мережею за допомогою контролера SDN, Cisco OpFlex прагне впровадити та визначити ці політики з мінімальним використанням контролера.

Зокрема, OpFlex прагне вдосконалити політику, щоб усунути масштабованість контролера та безпосередньо керувати каналом зв'язку, не перетворюючись на вузьке місце в мережі, просуваючи таким чином певний рівень інтелекту до пристроїв. Стандарт дозволяє визначати політики в логічному централізованому сховищі в контролері SDN, таким чином, що OpFlex може обмінюватися даними та застосовувати відповідні політики в підмножині розподілених елементів політики на комутаторах. Стандарт дозволяє двонаправлене повідомлення політик, мережевих подій та статистичної інформації моніторингу. Надання інформації в режимі реального часу може, в свою чергу, використовуватися для внесення коригувань в мережу. Розглянуті комутатори містять агент OpFlex, що підтримує стандарт Cisco OpFlex. Деякі гіганти галузі, включаючи Microsoft, IBM, F5, Citrix та Red Hat, продемонстрували прихильність до впровадження агента OpFlex у свою лінійку продуктів. OpFlex покладається на традиційні та розподілені протоколи управління мережею, щоб надсилати команди до вбудованих агентів у комутаторах. Однією з головних причин ранньої адаптації OpenFlow був рівень контролю, який він може запропонувати розробникам для розробки додатків для управління мережею з мінімальною підтримкою з боку постачальників мереж. Щоб стандартизувати OpFlex, Cisco подала протокол до процесу стандартизації IETF, і в даний час кілька постачальників працюють над стандартизацією, а також з метою посилення прийняття стандарту.

Стандарти зв'язку «півночі»

Репрезентативний стандарт державного трансферу (RESTful) (1996):

RESTful дотримується архітектури програмного забезпечення, розробленої для консорціуму World Wide Web (W3C), що охоплює всі комунікації клієнт-сервер. Спочатку стандарт був введений [18]. Його головна мета - запропонувати масштабованість, загальність та незалежність, дозволяючи включення проміжних компонентів між клієнтами та серверами для полегшення цих необхідних функціональних можливостей. Як клієнти, так і сервери можуть розроблятися самостійно або в тандемі різних постачальників.

У RESTful серверний компонент не має стану, і клієнти відстежують свої окремі стани, щоб забезпечити масштабованість. Відповіді сервера можна кешувати протягом певного часу. Кожну сутність або глобальний ресурс можна ідентифікувати за допомогою глобальних ідентифікаторів, таких як URI, і він може реагувати на створення, читання, оновлення та видалення (CRUD) операцій. Єдиним інтерфейсом для кожного ресурсу є GET (читання), POST (Вставка), PUT (запис) та DELETE (видалення). Типи даних можуть визначати такі мережеві компоненти, як контролер, правило брандмауера, топологія, конфігурація, комутатор, порт, посилання та обладнання. RESTful є поширеним у більшості архітектур контролерів як вибірковий API на вибір разом з Java API. Однак одним із головних недоліків RESTful є відсутність публічної підписки або поточного каналу, що інформує програму / послугу про зміни в мережі. Як і HTTP, REST не повідомляє, коли сторінка змінилася, і вимагає частого оновлення. Тому розробники додатків періодично використовують виклики циклу для отримання та подальшого опублікування оновлень до окремих комутаторів на основі заздалегідь визначених політик. Підтримка RESTful включена майже у всі основні контролери SDN, включаючи Ryu [19] та OpenDayLight [20], а також кілька власних платформ постачальників. Згідно з [21], REST API став найпоширенішим вибором для NBI в SDN, оскільки він є дуже розширюваним і підтримуваним для управління послугами як з даних, так і з планів управління.

Стандарт ініціативи відкритих шлюзів (OSGi) (2000): Стандарт OSGi включає набір специфікацій для динамічного складу додатків із використанням багаторазових компонентів Java, званих пакетами [22]. Пакети публікують свої послуги в реєстрі служб OSGi, щоб знаходити та використовувати служби інших пакетів, які можна встановити, запустити, зупинити, оновити та видалити. Модулі визначають, як пакет може імпортувати та експортувати код. Рівень безпеки обробляє середовище безпеки та виконання визначає методи та класи, доступні на певній платформі. Пачка може або отримати послугу, або прислухатися до появи або зникнення послуги. Кожна послуга має властивості,

що дозволяють іншим службам вибирати серед безлічі пакетів, що пропонують одну і ту ж послугу. Послуги є динамічними, і група може вирішити скасувати свою послугу, що може призвести до того, що інші групи припинять використовувати конкретну послугу.

Стандарт модельованого рівня абстрагування послуг (MD-SAL) (2014): Проект у [23] розробляє контейнери додатків, які можуть бути побудовані на вершині OSGi, щоб спростити операційні аспекти упаковки та встановлення програм у контролері OpenDayLight. Для подальшого сприяння розробці додатків модельований підхід MD-SAL у [24] абстрагує деякі служби, що використовуються в контролері OpenDayLight, і пропонує розробникам та адміністраторам мережі можливість уніфікувати API на північ та на південь із структурами даних багатьох служб та окремі компоненти контролера. Сама структура даних описується мовою Yan Another Next Generation (YANG), яка використовується для моделювання служби та абстракцій даних як єдиної системи [25]. YANG може моделювати семантику та організацію даних, включаючи дані конфігурації та / або операцій як дерево. API контролера на північ використовує дані самоопису в XML, що спрощує розробку додаткових керованих контролером функціональних можливостей, а також додатків SDN. Модулі, що забезпечують певні функціональні можливості мережі, можуть визначити схему, яка дозволяє інтерпретувати структури даних через MD-SAL простим способом. MD-SAL використовує API для підключення та прив'язки запитів та послуг та забезпечує додатковий рівень, що містить всю необхідну логіку для прийому та делегування запитів [26]. Сама архітектура SAL включає підсистему верхнього рівня, що складається з компонентів контролера або додатків, які використовують SAL контролера для зв'язку з іншими компонентами та плагінами контролера (сховище даних, валідатор тощо) та (ii) вкладені підсистеми, які відкривають набір функціональних можливостей і може мати кілька екземплярів, приєднаних до загальної системи (елементи мережі, віртуальні системи тощо). Підхід MD-SAL є відносно агностичним, підтримуючи будь-яку модель обслуговування. Крім того, схема стикує

горизонтальні модулі і дозволяє розробникам використовувати загальні інтерфейси для виявлення послуги та подальшого використання.

Приближення до мережевих підходів (2015): Інтерфейс, що базується на намірах (INBI), представляє еволюцію від статичної до динамічної настройки мережі за допомогою простої специфікації намірів програми, запрограмованої контролером у фізичні та віртуальні пристрої. INBI дозволяє програмам SDN описувати свої наміри, не обов'язково вказуючи методи їх досягнення. Контролер SDN, у свою чергу, перетворює ці наміри в команди конфігурації низького рівня в площині даних для подальшого виконання. Тому додаток або користувач не звертає уваги на базову конфігурацію інфраструктури, що забезпечує додаткову гнучкість та автоматизацію для розробників додатків та забезпечує гнучке розгортання служб. З намірами INBI перед нами стоїть вагомий випадок впровадження в SDN із багатьма перевагами, такими як масштабованість додатків, більша портативність, підвищена узгодженість, контекстне управління. Однак для практичного розгортання INBI потрібна мова для перекладу намірів користувача або програми [27], отже, нещодавно основна увага приділяється розробці INBI на основі намірів для власних контролерів [28].

1.3. Особливості структурної організації системи IoT на основі SDN-мереж

На даний час SDN може бути використаний як сполучення для адаптації IoT до реального світу. На рис. 1.3 розглядається архітектура IoT на основі SDN. Будь-який об'єкт в IoT може взаємодіяти з будь-яким іншим об'єктом через мережу, що підтримує SDN. Кожен пристрій IoT має агент IoT, який взаємодіє з контролером IoT.

Агент IoT відповідає за сприйняття, аналіз та збір даних із середовища для досягнення цілей користувача. Всі агенти IoT повинні бути зареєстровані в контролері IoT з такими деталями, як його ідентифікатор об'єкта, адреса, мережевий протокол зв'язку та основна мережа.

Контролер IoT прийме необхідні рішення на основі даних, наданих йому агентами IoT. Ці рішення відображаються на базовій фізичній мережі через контролер SDN. Контролер IoT при отриманні запиту на підключення від свого агента IoT будує правила переадресації залежно від використовуваних мережевих протоколів і передає ці правила контролеру SDN.

Після того, як контролер IoT отримає адресу або ідентифікатор об’єктів призначення, йому потрібно знайти його в мережі. Це легко, оскільки агенти IoT будуть зареєстровані

з контролером IoT, і вони надають відповідний ідентифікатор або адресу.

Контролер SDN встановлює мережевий шлях, який з'єднує обидва об'єкти, запускаючи алгоритм маршрутизації з інформацією про топологію як з IoT, так і з рівня SDN.

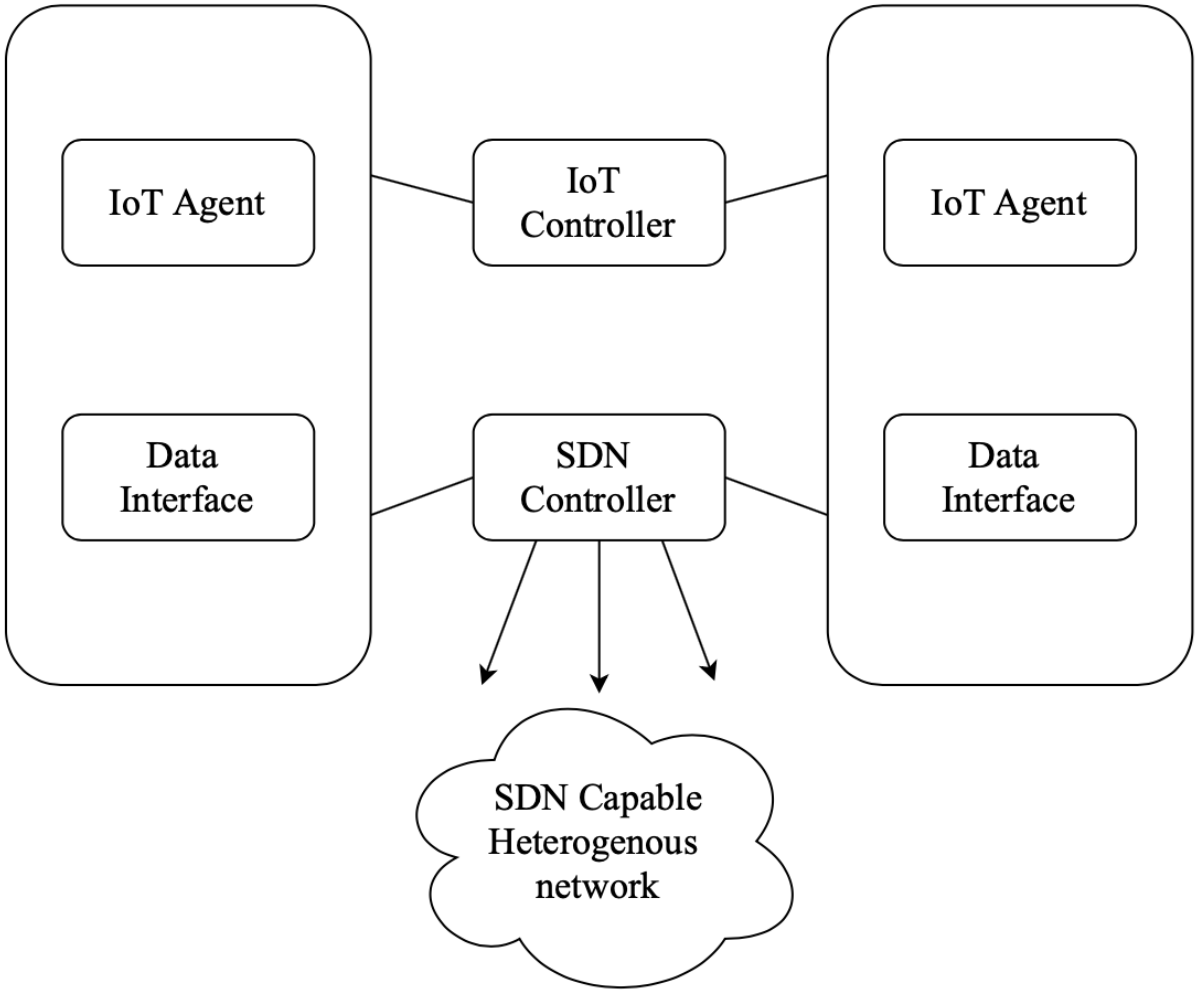


Рис. 1.3 – SDN архітектура для IoT

Загальний вигляд інтеграції SDN та IoT, як показано на рис. 1.3, включає мінімальний набір функціональних блоків, диференційованих за агентом та площиною, до якої вони належать, тобто об'єктом або мережею, даними або площиною управління. Таким чином, два об'єкти, підключені до мережі із підтримкою SDN, зможуть взаємодіяти з контролером IoT, використовуючи свої внутрішні агенти IoT. Мета полягає в тому, щоб надати контекстну інформацію контролеру для прийняття необхідних рішень та відображення їх у базовій мережі. Хоча контролер IoT зображений лише як один функціональний блок, він внутрішньо модульний, тому до накладеного IoT можна додати нову функціональність, не впливаючи на інші елементи, не потребуючи встановлення нових зв'язків з контролером SDN.

У поточних мережевих архітектурах і протоколах звичайне спілкування починається, коли мережевий об'єкт, запитувач, просить мережу надіслати пакет даних або повідомлення певного типу іншому мережевому об'єкту, відповідачу. Перш ніж це може відбутися, запитувач повинен знати ідентифікатор або адресу відповідача, і він повинен бути вказаний у транзакції. Ця модель застосовується до більшості протоколів, включаючи протоколи IoT.

Для такого спілкування мережа встановлює шлях від запитувача до відповідача. Такий шлях може бути логічним (встановлення постійного з'єднання або схеми) або віртуальним (просто слідуючи таблицям маршрутизації). Тут SDN входить у гру, щоб дозволити об’єктам, які покладаються на різні (і, отже, неоднорідні) протоколи, спілкуватися між собою. Таким чином, механізми SDN можна використовувати для встановлення шляху, який з'єднує обидві кінцеві точки. Тут це називається шляхом пересилання, і це досягається встановленням необхідних правила пересилання для всіх елементів пересилання, знайдених у такому шляху.

На цьому етапі контролер IoT знаходить свою роль. Він повинен отримати інтерес до зв'язку від запитувача, знайти відповідача на графіку мережі, розрахувати шлях, використовуючи якийсь алгоритм маршрутизації, побудувати правила переадресації залежно від природи протоколів, що

використовуються об'єктами, і, нарешті, повідомити такі правила до Контролера SDN для встановлення їх в пересилачах.

Комунікаційні інтереси надсилаються на контролер IoT агентами IoT, встановленими в об’єкти. Вони будуть інтегровані з іншими компонентами об'єкта, щоб з'ясувати необхідні деталі встановленого зв'язку, наприклад, ідентифікатор або адресу об'єкта призначення. Ця інформація надсилається контролеру IoT до того, як зв’язок фактично ініціюється, тому мережа буде підготовлена до її продовження.

Як тільки контролер IoT отримує такий інтерес з ідентифікатором або адресою респондента, він повинен знайти його на графіку мережі. Це легко, оскільки агенти IoT будуть зареєстровані в контролері IoT, і вони нададуть відповідний ідентифікатор або адресу. Потім цей контролер обчислює шлях, який з’єднує обидва об’єкти, запускаючи алгоритм маршрутизації з інформацією про топологію як на рівнях IoT, так і на рівні SDN. Таким чином, ми впевнені, що отриманий шлях вписується в мережу і дотримується можливих правил, встановлених адміністраторами мережі на рівнях IoT або SDN.

Знаючи конкретні протоколи, що використовуються обома кінцевими точками зв'язку, і розрахувавши шлях, по якому буде йти інформація, контролер IoT тепер будує необхідні правила пересилання для кожного елемента шляху. Залежно від протоколу, що використовується об’єктами, правила матимуть різні поля відповідності від пакетів і матимуть різні дії. Якщо протоколи різні і несумісні, деякі правила змусять експедиторів адаптувати або перекласти це так, щоб пакети доходили до місця призначення таким чином, щоб адресат зрозумів.

Нарешті, отримані правила передаються контролеру SDN, який використовує відповідний протокол управління, щоб встановити їх у пересилачах, зазначених контролером IoT. У цей момент шлях побудований, і обидва об’єкти можуть розпочати розмову. Весь процес дуже швидкий, але може спричинити деяку затримку. Однак він вводиться лише до першого

пакету, оскільки після встановлення правил у пересилачах вони працюють майже настільки ж ефективно, як будь-який комутатор або маршрутизатор, особливо якщо комутатори SDN побудовані на конкретному та оптимізованому обладнанні.

Як зазначалося вище, для того, щоб сформувати частину накладання IoT, кожен об'єкт повинен бути зареєстрований у контролері IoT. Це завдання виконує агент IoT, пов'язаний з об'єктом. Процес реєстрації надасть контролеру IoT необхідну інформацію про об'єкт, наприклад, його мережевий протокол, основну мережу, до якої він прив'язаний, та його ідентифікатор або адресу.

Контролер SDN може не тільки керувати неоднорідною мережею IoT, але також може контролювати вхідний та вихідний трафік. Контролер SDN може ефективно захистити мережу IoT від атаки всередині та зовні.

Було проведено багато дослідницьких робіт для забезпечення безпеки шляхом впровадження брандмауерів, IPS та IDS поверх контролера SDN [13], [29], [30]. Політики безпеки встановлюються на комутатори OpenFlow. Однак аутентифікація мережевих пристроїв, користувачів та об'єктів, що підключаються до користувачів за допомогою обох неоднорідних технологій в IoT, ще не відбулася.

Рішення з використанням одного єдиного контролера SDN: Структури безпеки, засновані на одному контрольованому SDN, обговорювались у попередніх роботах [31], [32], [33], [34]. Атака відмови в обслуговуванні (DOS) може статися на контролері SDN, який є єдиною точкою відмови. Якщо відбувається внутрішня атака і порушує контролер SDN, тоді можна взяти повний контроль над мережею.

Рішення з використанням декількох контролерів SDN: У роботі [35] спостерігається, що кілька контролерів можуть підвищити надійність та стійкість до несправностей. Коли один із контролерів SDN виходить з ладу, інший контролер може взяти на себе управління, щоб подолати збої системи. Але спостерігається зниження продуктивності мережі з кількома контролерами. Оскільки кожен контролер має частковий вигляд мережі, контролерам потрібно

співпрацювати між собою та обмінюватися інформацією, створюючи накладні витрати.

1.4. Проблеми безпеки в сфері IoT

Щоб зробити IoT безпечним, слід створити надійну адаптацію до загальних протоколів, що використовуються в мережевому середовищі IoT. Роль IP для Інтернету є значною і пропонується як рішення для Інтернету речей, особливо з розвитком IPv6. Однак у реальному світі цей підхід має багато проблем та недоліків, пов'язаних з неоднорідністю пристроїв та мереж, що беруть участь в IoT.

Ці об'єкти та їх протоколи відповідають конкретним моделям для досягнення конкретних цілей користувача. Спроба вписати всі ці різноманітності об’єктів у загальний єдиний протокол - невдалий варіант.

З іншого боку, підхід, що визначається програмним забезпеченням (SDN), фокусується на програмованості всіх елементів мережі. У цьому процесі управління та площина даних розділяються в пристроях маршрутизації, завдяки чому функціональність проміжних мережевих пристроїв була спрощена до простого пересилання пакетів. Загальна площина управління визначає правила пересилання цих спрощених проміжних пристроїв.

SDN забезпечує перспективне майбутнє IoT з точки зору його реальної адаптації до реального світу.

Через неоднорідність та складність об’єктів та мереж в Інтернеті речей традиційні методи аутентифікації та авторизації можуть бути непридатними. Крім того, обмежені ресурсами пристрої в Інтернеті речей обмежують використання складного механізму захисту.

Ідентифікація об’єкта: Ідентифікація об’єкта є складною сферою в Інтернеті речей. Система доменних імен (DNS) надає послуги перекладу імен для користувачів Інтернету, які вразливі до атак, таких як атака отруєння кешу DNS, атака "людина посередині". Розширення безпеки служби доменних імен (DNSSEC) згідно з IETF RFC4033 розгортається як розширення безпеки DNS

[13]. Але все ще складно розгорнути DNSSEC в IoT через високі накладні витрати на зв'язок.

Конфіденційність та цілісність: Дані, що сприймаються різними фізичними вузлами в Інтернеті речей, потрібно збирати та анонімізувати. Потрібно виконати шифрування та дешифрування даних. Пристрої, обмежені ресурсами, такі як вузли датчиків в IoT, не здатні виконувати такі складні криптографічні операції безпеки, створюючи лазівки в конфіденційності та цілісності даних.

Аутентифікація та авторизація: Традиційні криптосистеми з відкритим ключем не можуть поміститися в екосистемі IoT. Аутентифікація та авторизація за допомогою криптографічно спільних ключів не застосовується. Швидко зростаюча кількість об’єктів зробить управління ключами важким завданням в Інтернеті речей. Відсутність глобального центру сертифікації в IoT є основною перешкодою. Криптографічні алгоритми, як правило, важкі і вимагають величезних відбитків пам'яті, що обмежує їх використання в пристроях з обмеженою пам'яттю в IoT.

Шкідливе програмне забезпечення в IoT: Сканування в режимі реального часу, яке виконується антивірусним програмним забезпеченням, є значними витратами на пристроях IoT. Symantec підтвердив існування першого шкідливого програмного забезпечення IoT, Linux, Darlloz, що ознаменувало введення проблеми з шкідливим програмним забезпеченням для безпеки IoT. Функціональність сканування в реальному часі антивірусу може призвести до недоступних накладних витрат на пристрої IoT. Загроза через зловмисне програмне забезпечення в Інтернеті речей є великою проблемою і відкритим напрямком досліджень.

Висновки до розділу 1

У розділі проведений огляд загальної концепції архітектури IoT на прикладі системи розумного будинку та деякі аспекти безпеки даної сфери, що підтверджує потребу в належному захисті системи на основі інтегрування SDN-мереж.

РОЗДІЛ 2
ВИБІР І ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Аналіз існуючих методів захисту інформації в системі

Побудова архітектури, описаної в попередньому розділі, розкриває низку дослідницьких питань, які повинні бути вирішені за допомогою проекту механізму інтеграції та архітектури загалом. Вони узагальнені таким чином:

- Спільна та неоднорідна схема ідентифікації: Об'єкти можуть використовувати різні схеми ідентифікації, в основному пов'язані з основним мережевим протоколом, який вони використовують. Для того, щоб дозволити їм взаємодіяти, нам потрібно вивчити різні підходи до ідентифікації, що використовуються протоколами, які будуть підтримуватися мережею IoT. Це визначить форму загального посилання на відображення, яке буде використовуватися контролером IoT для проектування кожного простору імен на інші, щоб ідентифікатори, що використовуються об'єктами, сумісні з їх протоколами;
- Де створити екземпляр агентів IoT: Хоча агенти IoT мають мало роботи, тому вони легкі, не всі об’єкти можуть мати можливість створити екземпляр агента самостійно. Тим не менше, існують інші місця, де вони можуть бути створені від імені цих об'єктів, такі як мережевий шлюз, до якого вони підключені, пересилач SDN (комутатор) або навіть разом з контролером IoT як інший модуль. Ці альтернативи повинні бути оцінені з різні точки зору, щоб найкращий з них був включений у проект або навіть зробити висновок, що потрібно враховувати більше одного;
- Алгоритм маршрутизації: Є багато хороших алгоритмів маршрутизації, які можна використовувати для пошуку шляху між двома об’єктами, але вони працюють лише з однією топологією. Описана тут архітектура вимагає алгоритму маршрутизації, який враховує стан двох окремих, але дублюючих топологій, базової топології SDN та топології IoT, що накладається. Алгоритм також повинен враховувати різні аспекти, такі як політики або пропускна здатність. Необхідно знайти належний

алгоритм для вирішення цієї вимоги та оцінити, щоб визначити, як він відповідає цілям цього підходу;

- Формулювання правила пересилання: Після того, як шлях обчислений, контролер IoT повинен відобразити його в різних правилах, які будуть надіслані різним пересилачам (комутаторам). Хоча це легко зробити, формулювання не є прямим і має враховувати різні адаптації, зіставлення, базові протоколи, поля відповідності та загальні дії, необхідні для належного пересилання пакета до місця призначення;
- Стабілізація Nourthbound API: Поточні моделі SDN включають так званий API на північ, який може використовуватися зовнішніми модулями, такими як контролер IoT, для передачі операцій управління контролеру SDN. Однак цей API недостатньо чітко визначений і не стабілізований, тому вимога поточного підходу полягає у пошуку (або дочеканні) стабілізації такого інтерфейсу;
- Модульність контролера IoT: У попередньому розділі ми описали основну функціональність контролера IoT, але він повинен бути відкритий для майбутніх удосконалень у вигляді нових модулів. Це вимагає ретельного вивчення найширших і довговічних модульних систем та того, як вони розвивалися в часі, тому обрана альтернатива забезпечує еволюційну здатність запропонованого підходу;
- Процедура розгортання: Початкова процедура розгортання повинна бути розроблена та проаналізована з точки зору SDN та наявних на даний момент інфраструктур. Оскільки це ключовий аспект успіху пропозиції, важливо перевірити її на основі поточних та майбутніх мереж на початкових етапах дослідження.

Ця частина досліджує деякі обмеження у впровадженні мереж IoT для вдосконалення різних сфер діяльності; ці обмеження можна подолати, застосувавши SDN для реалізації поняття "Програмне забезпечення визначає Інтернет речей". Розрізняється:

- Ефективне управління мережею: потужність технологій IOT дозволить підключити мільярди речей до Інтернету протягом декількох років, що призведе до генерування великих обсягів даних, які повинні бути оброблені оптимально та ефективно, керуючи всіма цими пристроїв та великий обсяг даних, які вони генерують, - це завдання, до якої потрібно поставитися серйозно. Технологія SDN дозволяє оптимально контролювати та розподіляти потоки трафіку з мінімізацією затримки та балансування навантаження в мережі, використовуючи глобальний вигляд мережі та централізоване управління даними. Застосування SDN у мережах IoT дозволить ефективно використовувати пропускну здатність шляхом балансування навантаження в мережі та більш точного передавання повідомлень.
- Мобільність: Як уже зазначалося, найближчі мільярди пристроїв будуть підключені до Інтернету речей у найближчі кілька років, і користувачі повинні мати можливість використовувати різні функції своїх пристроїв та отримувати доступ до мережі IoT до будь-якого час і з будь-якої області без будь-яких проблем. Технологія SDN дозволяє керувати цими пристроями, що дозволяє безперешкодно обробляти мобільність користувачів.
- Виправити інтенсивне використання мережевого ресурсу: зловживання мережею користувачами збільшує навантаження на мережеве обладнання, що зменшує продуктивність останнього, для підвищення ефективності роботи мережі необхідно скласти план якість обслуговування, про яке користувачі заздалегідь запитують, та налаштування мережевої інфраструктури для оптимізованої інженерії трафіку. Використовуючи SDN, контролер SDN використовує глобальний погляд на мережу, щоб визначити оптимальний шлях та правила, які повинні застосовуватися до потоку даних, що надходить у мережеве обладнання, що дозволяє споживати менше мережевих ресурсів і для підвищення його ефективності.

- Оптимізація управління енергією: величезна маса даних буде генерована мільярдом пристроїв, які підключатимуться до мережі IOT, що включає велику кількість центрів обробки даних для обробки всієї цієї інформації, буде спожито велику кількість електричної енергії для живлення всіх цих центрів обробки даних для цього потрібно створити енергоефективні центри обробки даних з інтелектуальною системою управління енергією, щоб мінімізувати їх споживання. Технологія SDN буде динамічно вмикати або вимикати обладнання центрів обробки даних на основі використання шляхом ефективного картографування мережевого трафіку через контролер SDN для правильного сервера, створюючи енергоефективні центри обробки даних IoT на основі SDN.

2.2. Опис елементів організації структури IoT на основі SDN

Ключовою абстракцією парадигми SDN є розділення площин управління мережею та переадресації. Концептуально в мережах SDN ресурси трактуються як динамічний набір довільно підключених пристроїв переадресації з мінімальним інтелектом. Логіка управління реалізована поверх так званого контролера SDN. Рисунок 2.1 ілюструє репрезентативну структуру типового контролера SDN. Точніше, контролер - це програмна платформа для тестування та перевірки поведінки перемикання/управління, зосереджуючись на API протоколів, що стосуються півдня. Контролери SDN розглядаються як логічно централізовані сутності, що відповідають за набір завдань, включаючи вилучення та підтримку загального уявлення про топологію та стан мережі, а також за створення логіки переадресації, що відповідає певному сценарію програми. На практиці контролер SDN управляє підключеннями до всіх комутаторів субстрату, використовуючи протокол, що спрямований на південь, такий як OpenFlow, і встановлює, модифікує та видаляє записи переадресації в таблиці переадресації підключених комутаторів за допомогою контрольних повідомлень, специфічних для протоколу.

Контролери SDN представляють центральну точку в системі SDN для нагляду та управління потоком трафіку між південними комутаторами/

маршрутизаторами (мережево) за допомогою API відповідно до вимог кожного додатка. Для того, щоб контролер SDN виконував такі завдання, йому потрібна певна інформація щодо стану наданої базової мережі колекціями модулів, що підключаються, які виконують різні процедури збору інформації про пристрої рівня зв'язку, забезпечуючи повний опис мережі нижче, а також можливості пристроїв. Крім того, коли контролер SDN має повний огляд мережі, він додає розширення та комплекти для поліпшення можливостей контролера та забезпечує настроювані правила переадресації, створені з використанням алгоритмів, які аналізували інформацію та статистичні дані, зібрані з мережі.

Архітектурна структура контролерів SDN складається з трьох співіснуючих шарів:

- Рівень на південь, який здійснює зв'язок із мережевими пристроями за допомогою різних типів протоколів, таких як OpenFlow, OVSDB, NETCONF тощо;
- Північний рівень, де існують і розроблені додатки та послуги, використовуючи абстракцію мережі, що забезпечується центральним рівнем, і передаючи інформацію контролеру через API REST на північ;
- Центральний рівень, який забезпечує рівень абстракції для розробників додатків, дозволяючи їм розробляти додатки та послуги. Центральний рівень містить сховища даних (операційні та конфігураційні), де зберігається необхідний і поточний стан системи, а також забезпечує шину повідомлень, що дозволяє північному шару здійснювати зв'язок з південним шаром.

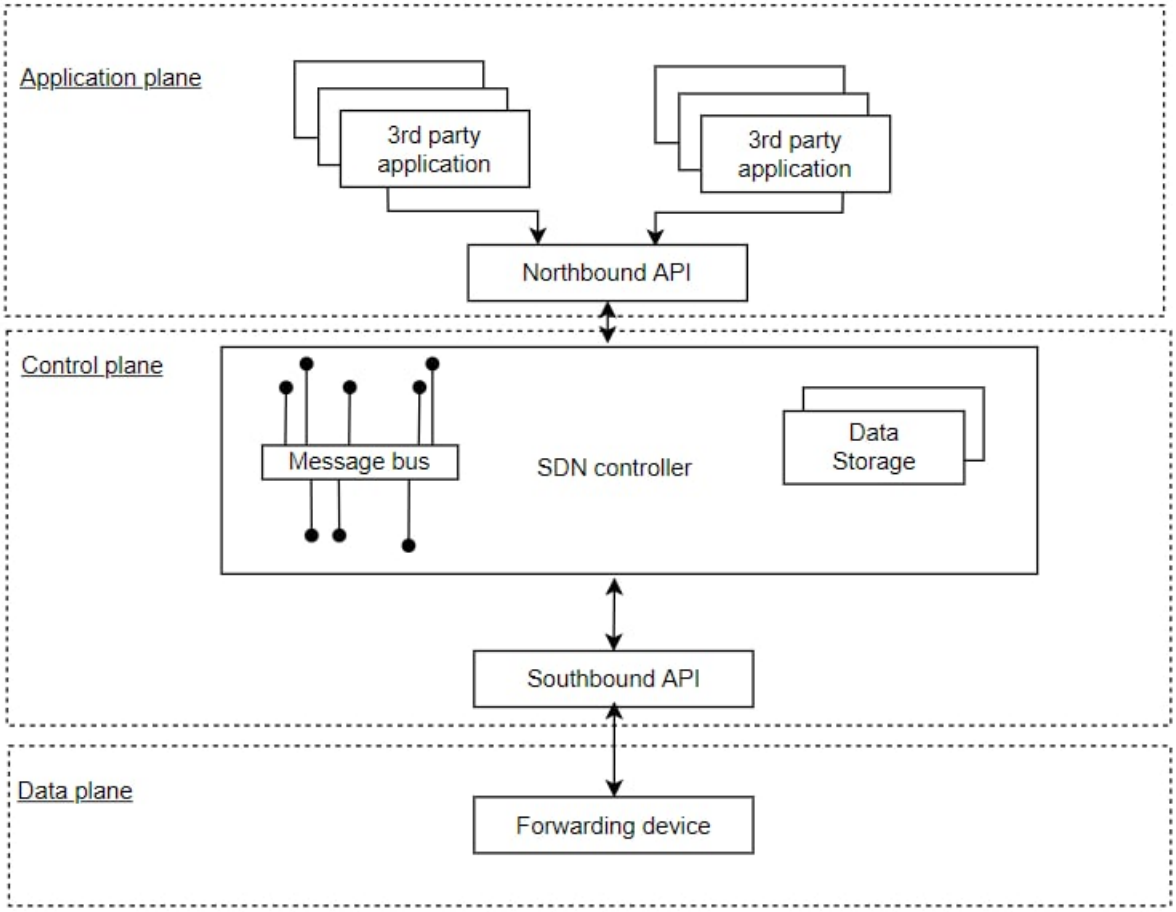


Рис. 2.1 – Структура SDN контролера

Комутатор OpenFlow - це програмна програма або апаратний пристрій, що пересилає пакети в середовищі SDN. Коли він отримує пакет, який не має потоку для (збіг + вихід порт), комутатор SDN зв'язується з контролером SDN (сервером), щоб запитати інформацію про те, як обробляти цей конкретний пакет. Потім контролер може завантажити потік на комутатор, можливо, включаючи деякі маніпуляції з пакетами, як показано на малюнку 2-4. Як тільки потік буде завантажений на комутатор, він буде переключати подібні пакети на швидкості дроту. Наявність центрального сервера, який обізнаний про макет мережі і може приймати всі рішення щодо переключення, при побудові шляхів маршрутизації дає системі SDN наступні переваги:

- контролер SDN може спрямовувати некритичний / масовий трафік на довші маршрути, які використовуються не повністю;

- контролер SDN може надіслати початкові пару пакетів на брандмауер, і як тільки брандмауер задоволений / приймає потік, контролер SDN може обійти брандмауер, тим самим знімаючи навантаження з FW і дозволяючи мультигігабітним центрам обробки даних замуровані;
- контролер SDN може легко реалізувати балансування навантаження також при високій швидкості передачі даних, просто направляючи різні потоки до різних хостів, лише виконуючи налаштування початкових потоків;
- мережевий трафік може бути ізольований без необхідності VLAN, контролер SDN може просто відмовити певним підключенням;
- спрощує налаштування точки доступу мережевого терміналу (TAP) для будь-якого порту або конкретного трафіку шляхом програмування мережі та надсилання повторюваних потоків на пристрої моніторингу мережі;
- комутація SDN дозволяє розробляти нові сервіси та ідеї для програмного забезпечення на контролері SDN.

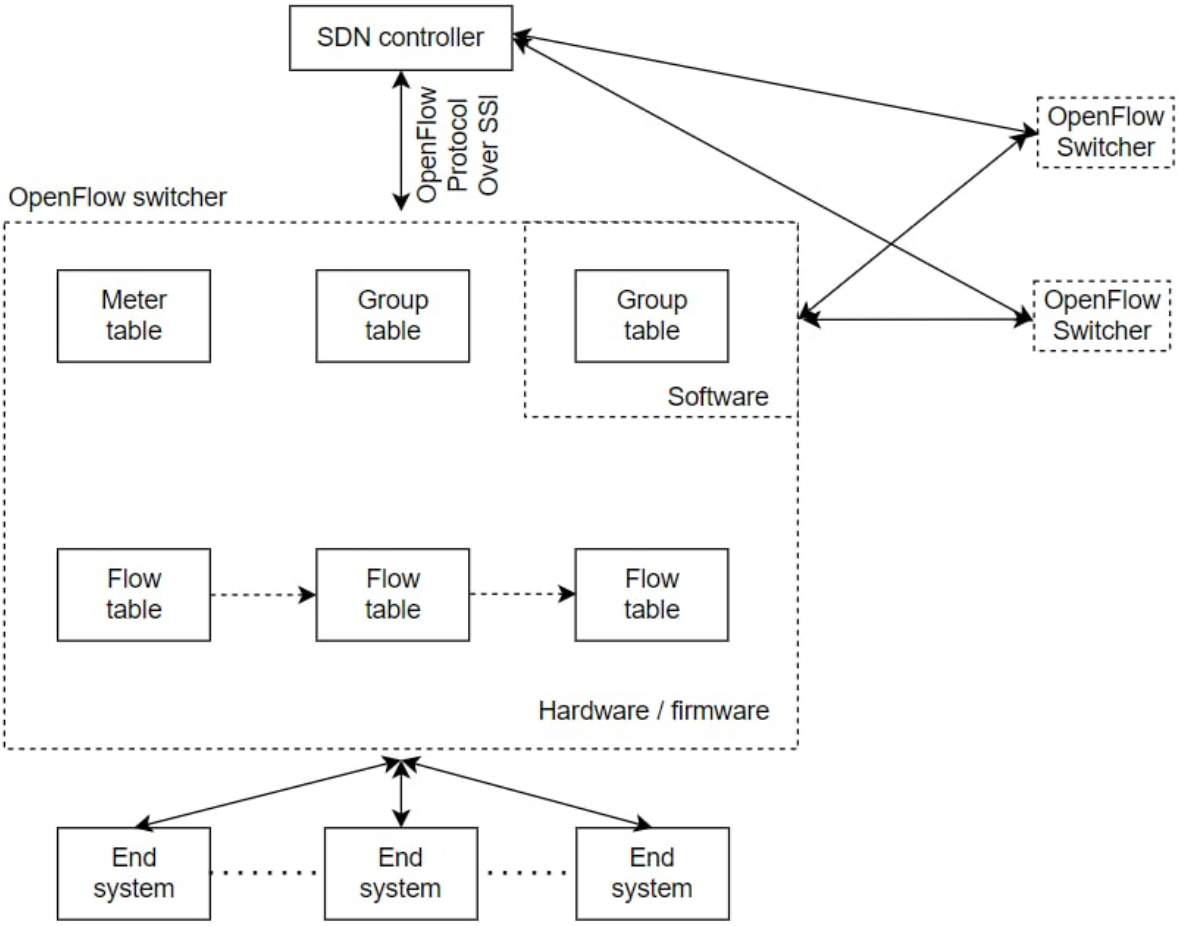


Рис. 2.2 – Структура SDN комутатора

У сучасному світі динамічних вимог стан мережі постійно змінюється та оновлюється відповідно до мінливих потреб кінцевих споживачів. Адміністраторам мережі та операторам потрібно налаштувати конфігурацію мережі відповідно. Традиційні маршрутизатори та комутатори мають на одному пристрої як управління (приймає рішення, пов'язане з управлінням трафіком), так і площину даних (фактичний механізм маршрутизації трафіку до пунктів призначення). Отже, ці пристрої стають повільними, дорогими, негнучкими, не масштабованими та липкими.

Оператори використовують зовнішні інструменти, мережеві сценарії для динамічної реконфігурації цих мережевих пристроїв. Це призводить до помилок конфігурації. Програмно-конфігурована мережева мережа SDN [35] була запропонована для вирішення цих проблем, з якими стикалися звичайні парадигми управління мережею.

Фонд відкритих мереж (ONF) [36] - це організація, яка займається просуванням та прийняттям SDN шляхом розробки відкритих стандартів. SDN - це інтелектуальна архітектура для програмованості мережі, що забезпечує абстракцію мережі. Він відокремлює площину управління від площини даних, щоб забезпечити програмованість мережі.

Як показано на рис. 2.3, SDN переміщує площину управління за межі перемикачів. Це дозволяє здійснювати зовнішнє централізоване управління даними через логічну сутність програмного забезпечення, відому як контролер SDN. SDN відокремлює програмне забезпечення від апаратного забезпечення. SDN централізує стан мережі на рівні управління. Це робить управління мережею, забезпечення, конфігурацію, оптимізацію ресурсів та безпеку мережі гнучким за допомогою автоматизованих програм SDN. Архітектура SDN включає набір API, що підтримує реалізацію загальних мережевих служб, таких як виявлення пристроїв, розподіл та відображення адрес, безпека, маршрутизація, контроль доступу, розподіл пропускної здатності та оптимізація ресурсів, управління споживанням енергії, підтримка зберігання, QoS та інші послуги, пов’язані з бізнесом.

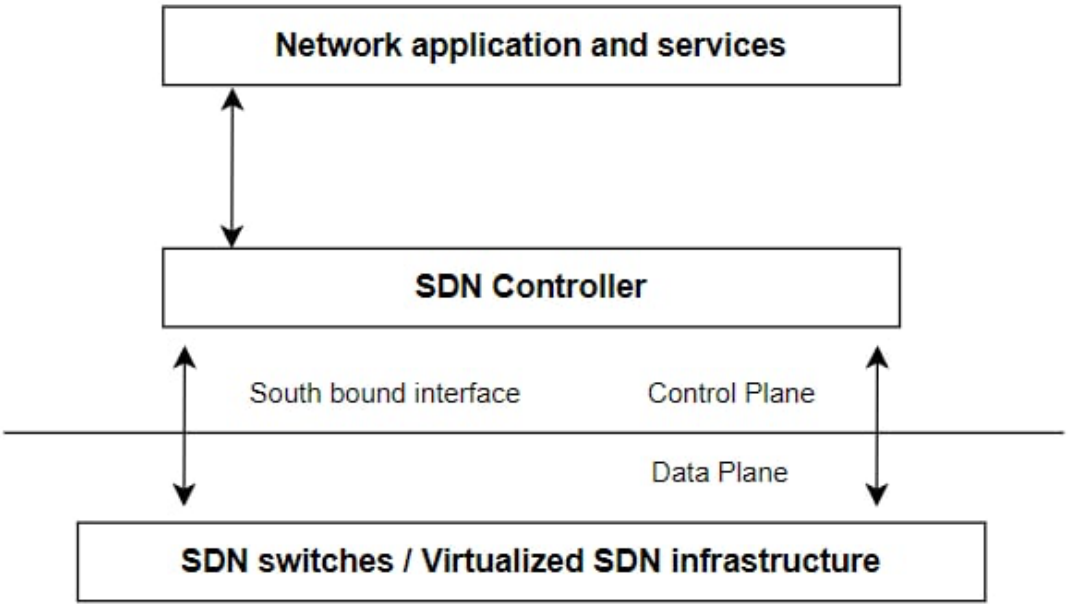


Рис. 2.3 – SDN-концепція

Відповідно до парадигми SDN, контролер SDN встановлює підключення до комутатора через протокол OpenFlow [37]. Площина управління виконується на окремому позашляховому обладнанні, такому як ПК, а також відомий як контролер відкритого потоку, як показано на рис. 2.4.

Контролер може оновлювати, додавати / видаляти записи потоку в таблиці потоків або реактивно у відповідь на пакети, або заздалегідь, використовуючи заздалегідь визначені правила. Інтерфейс контролера може бути виконаний на будь-якому постачальнику обладнання та операційній системі з високою продуктивністю. Площина даних виконується на німій (немає рівня Layer2, Layer3), потужних перемикачів, відомих як Open Flow Switch, для переадресації даних зі швидкістю передачі. Топологія може мати один контролер SDN, що керує одним або кількома комутаторами, або один комутатор, керований декількома контролерами.

SDN OpenFlow [38] забезпечує північний інтерфейс, який є взаємодією високого рівня між контролером та мережевим додатком / послугами. Інтерфейс на південь - це взаємодія нижнього рівня між контролером та пристроями, такими як комутатори.

В даний час SDN використовується в DCN [39], [40] для збору даних

про споживання смуги пропускання від вузлів, підключених до швидкої мережі в центрі обробки даних. Зібрана статистика включає також використання посилань.

У IoT-контролері потрібно збирати інформацію про стан із дуже неоднорідної мережі та розподіленої мережі. Дані про трафік IOT будуть чутливими до часу та даними реального часу. Має бути передбачено зменшення накладних витрат на збір. Затримка, тремтіння, втрата пакетів, пропускна здатність - це статистика, зібрана в IoT, на відміну від DCN.

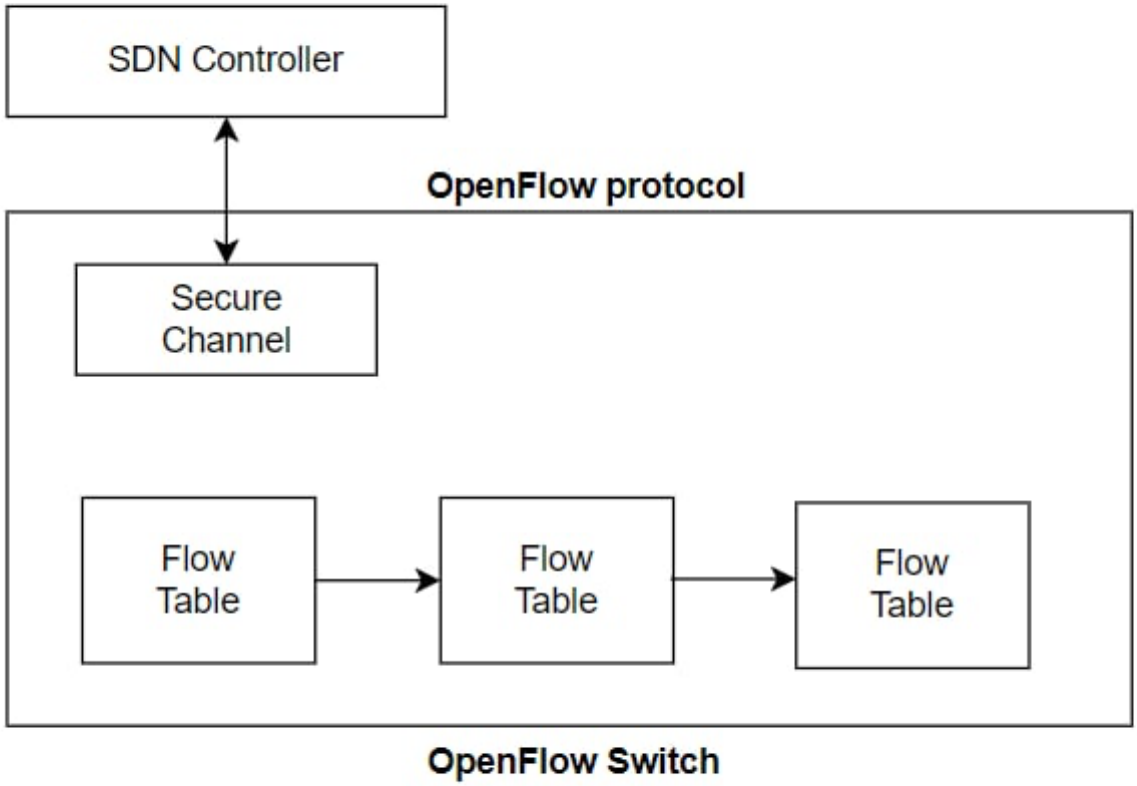


Рис. 2.4 – Структура SDN-комутатора

Висновок до розділу 2

У розділі проведений опис та обґрунтування предметів розробки схеми з підтримкою належного захисту IoT пристроїв на основі SDN-мереж, актуалізація методів захисту сфери IoT.

РОЗДІЛ 3

МОДИФІКОВАНИЙ АЛГОРИТМ СИНТЕЗА СТРУКТУРИ СИСТЕМИ IOT

3.1. Опис системи моделювання

Для моделювання хостів, комутаторів, посилань та контролерів SDN/OpenFlow був використаний Mininet, який[1] дозволяє створювати топології дуже великих розмірів до тисяч вузлів і виконувати тестування на них. Він має дуже прості інструменти командного рядка та API. Mininet дозволяє користувачеві легко створювати, налаштовувати, ділитися та тестувати мережі SDN. (Рис. 3.1)

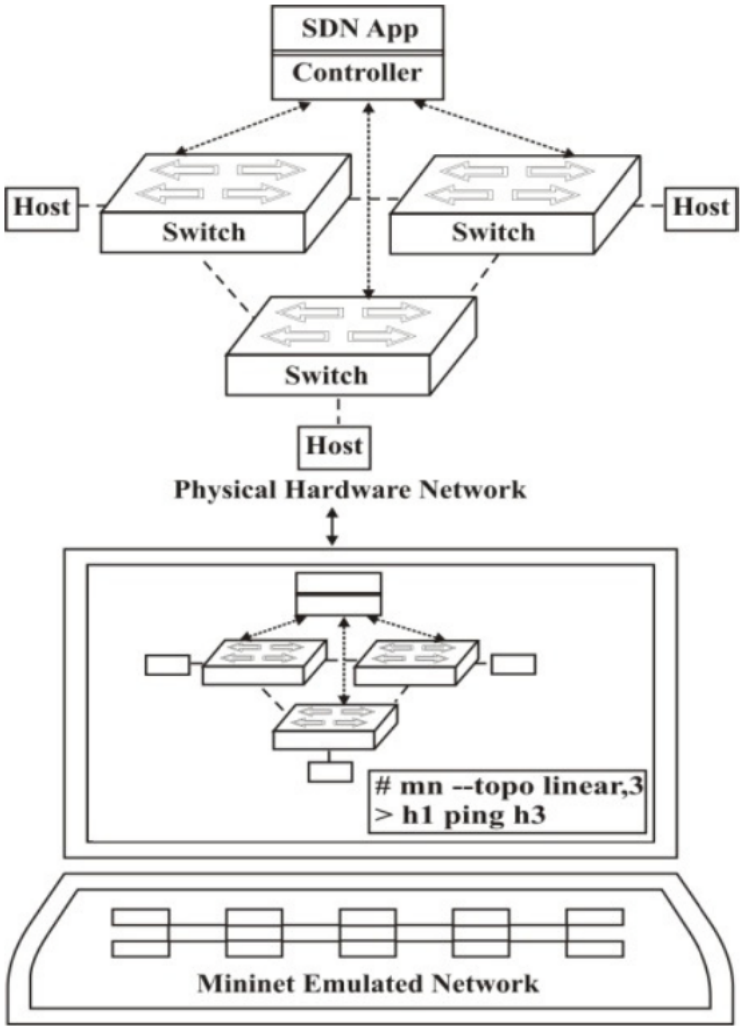


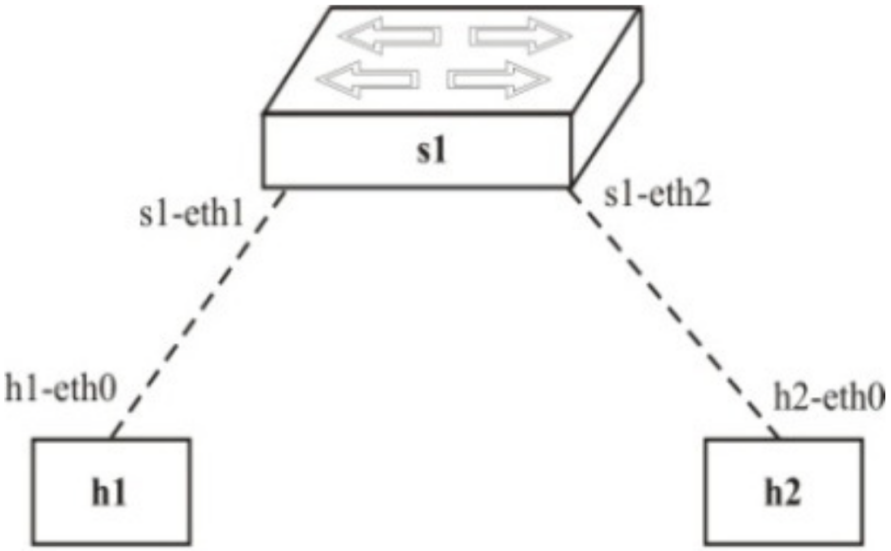
Рис. 3.1 – Емуляція топології мереж у Mininet

Mininet - це вільно доступне програмне забезпечення з відкритим кодом, яке імітує пристрої OpenFlow та контролери SDN. Mininet може імітувати мережі SDN, може запускати контролер для експериментів [2]. Це дозволяє емулювати реальні мережеві сценарії. Кілька контролерів SDN включені в Mininet VM. Контролери за замовчуванням хороші, але для реалізації попередніх концепцій використовується контролер POX [3].

Mininet містить кількість топологій за замовчуванням, таких як мінімальна, одиночна, зворотна, лінійна та дерево [10]. Цей розділ пояснює ці топології по черзі. Розуміння методу іменування інтерфейсів, хостів та комутаторів має важливе значення для процвітання за допомогою Mininet. Перемикачі називаються від s1 до sN. Хости називаються h1 - hN. Інтерфейси хосту називаються префіксом до імені хосту, за яким слідує ім'я Ethernet, починаючи з 0. Перший інтерфейс хосту «h1» називається «h1-eth0», а третій інтерфейс хосту «h2» називається «h2-eth2». Перший порт комутатора «s1» називається «s1-eth1». У комутаторах нумерація починається з 1.

1. Мінімальна - дуже проста топологія, яка містить 1 комутатор OpenFlow і 2 хости. Він також створює зв'язки між комутатором та двома хостами (рис. 3.2).

mn--topo minimal



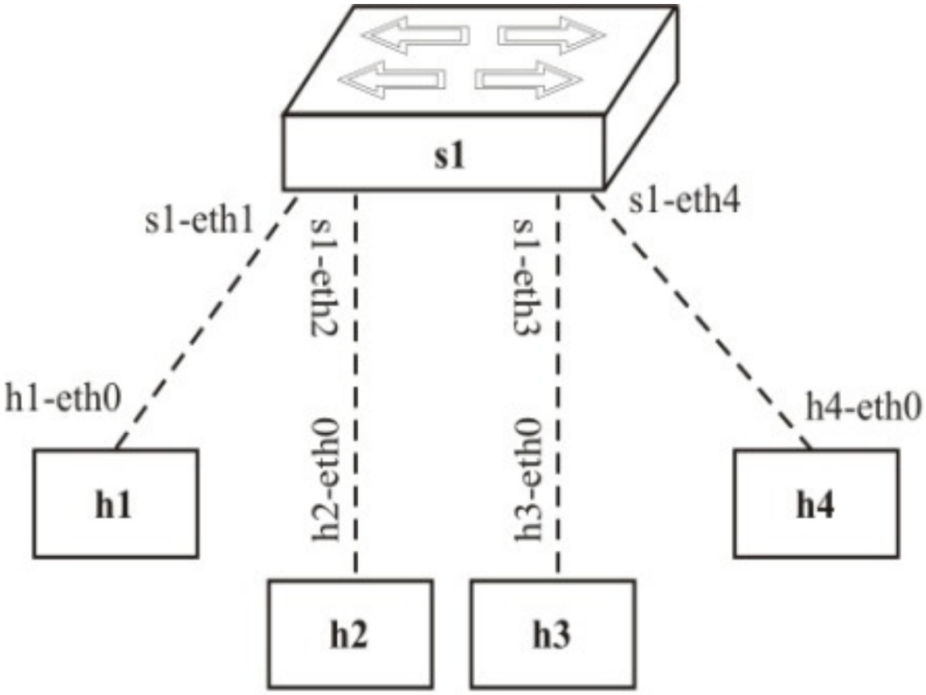

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

Рис. 3.2 – мінімальна топологія

2. Одинична

Це проста топологія з одним комутатором OpenFlow та k хостами. Це також створює зв'язок між комутатором та k хостами (рис. 3.3).

mn--topo single, 4

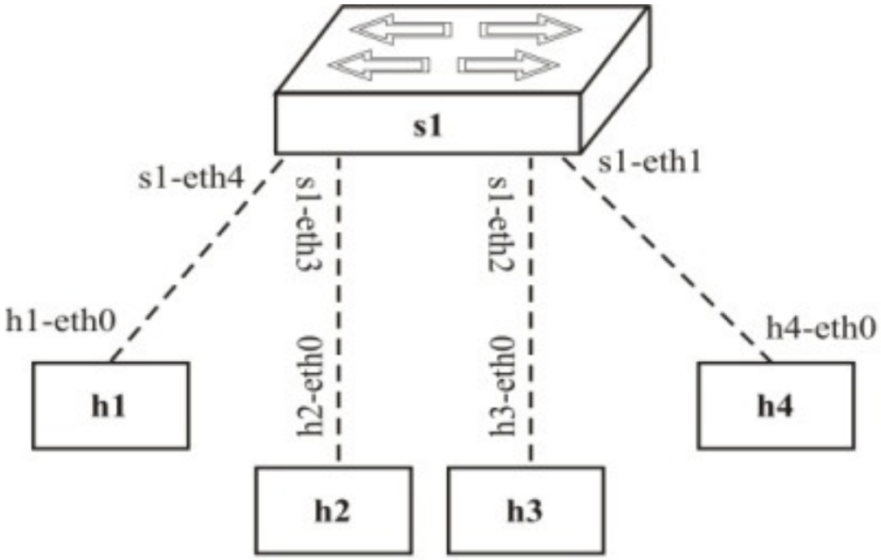



```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0
c0
```

Рис. 3.3 – Одинична топологія

3. Зворотня – схожа на одиночну топологію, але порядок з'єднання зворотній (рис. 3.4).

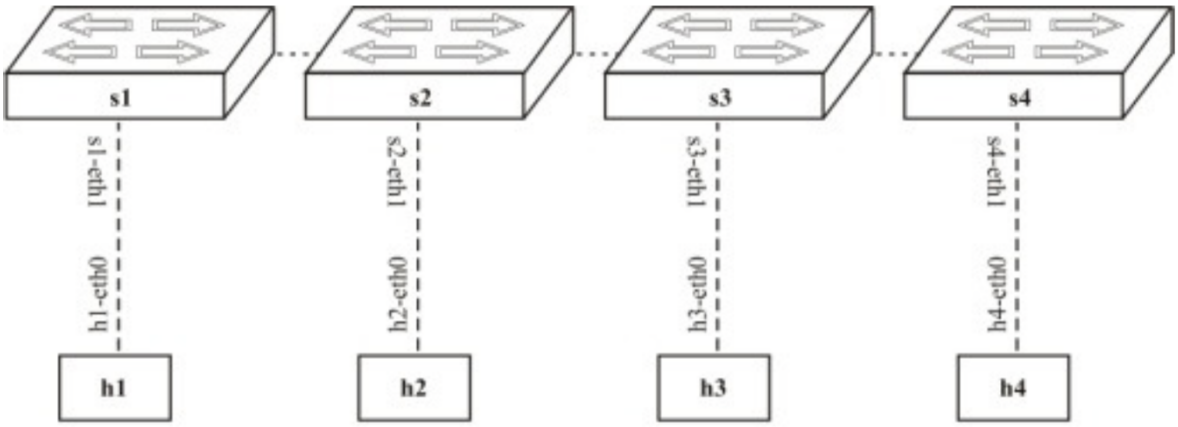
mn--topo reversed, 4




```
mininet> net
h1 h1-eth0:s1-eth4
h2 h2-eth0:s1-eth3
h3 h3-eth0:s1-eth2
h4 h4-eth0:s1-eth1
s1 lo:  s1-eth1:h4-eth0 s1-eth2:h3-eth0 s1-eth3:h2-eth0 s1-eth4:h1-eth0
c0
```

Рис. 3.4 – Зворотня топологія

4. Лінійна – топологія містить k комутаторів і k хостів. Він також створює зв'язок між кожним комутатором та кожним хостом та між комутаторами (рис. 3.5).




```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
```

Рис. 3.5 – Лінійна топологія

3.2. Розробка програмної частини

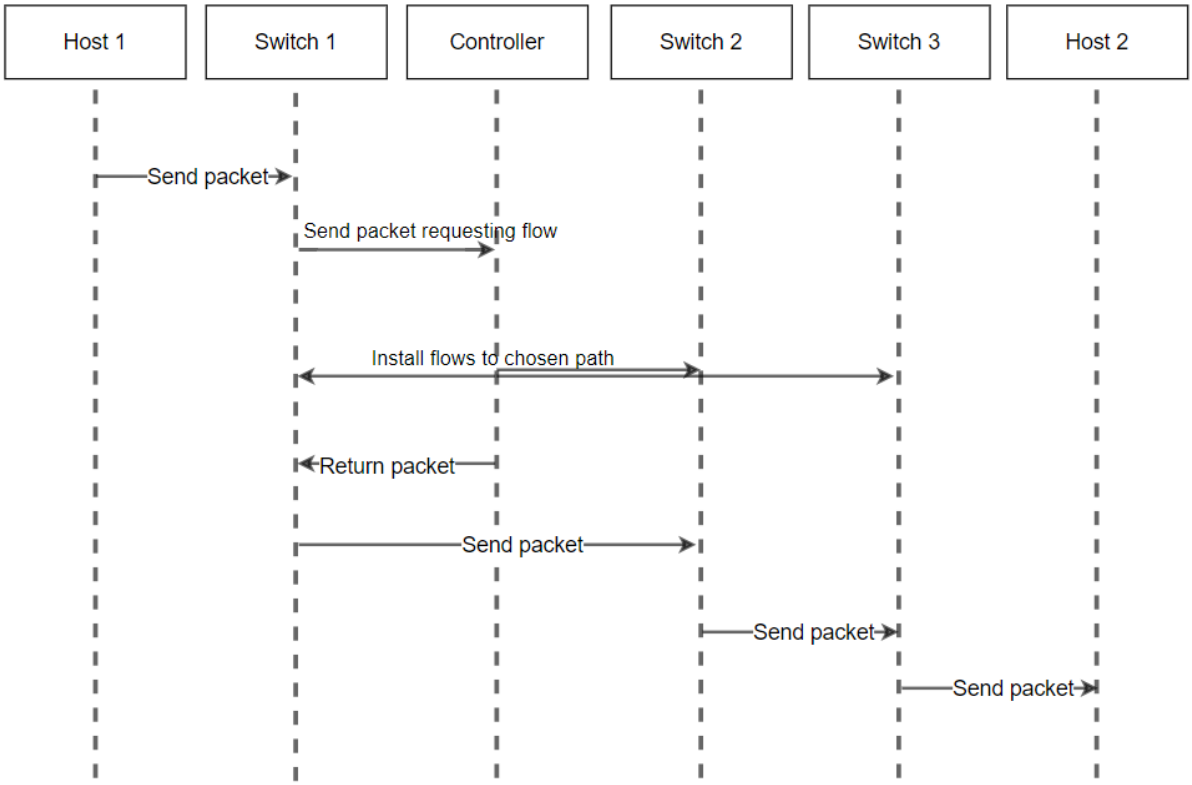


Рис. 3.6 – Обробка пакетів в OpenFlow

Для створення моделі програмно-конфігурованої мережі був використаний додаток Mininet, який є дуже потужним інструментом мережевої емуляції на основі Linux, який використовується багатьма дослідниками. Це інструмент, який дозволяє встановлювати певні параметри, такі як пропускна здатність, затримка, втрата пакетів та розмір черги для різних посилань у OpenV Switches, що використовуються в програмній мережі. Для моделювання програмно-конфігурованої мережі в OpenFlow (рис. 3.6) за допомогою емулятора Mininet були виконані наступні кроки:

- Створення програмно-конфігурованої мережі за допомогою команди:

```
$ sudo mn --topo tree,2,2 --controller=remote,ip=127.0.0.1,port=6633
```

- Команда `sudo mn --topo tree,2,2 --controller=remote` створює програмно-конфігуровану мережу з контролером, перемикачем OpenFlow і чотирма хостами.
- Перерахування вузлів, доступних у конфігурованій програмним забезпеченням мережі, за допомогою командних вузлів:

```
mininet> nodes
```

Результат команди показує:

доступні вузли: c0, h1, h2, h3, h4, s1, s2, s3

Хости h1, h2, h3 і h4 створюються, а контролер додається в програмно-конфігуровану мережу.

На малюнку 3.7 всі віртуальні хости підключені до комутаторів відкритого потоку, а комутатори відкритого потоку - до контролера POX. Контролер POX - це платформа, реалізована на python для емуляції площини управління в програмно-конфігурованій мережі. Комутатори OpenFlow використовуються для підключення хостів до контролера POX. У цій топології контролер налаштований на порт 6633 та адресу 127.0.0.1.

Логіка роботи топології реалізована на програмній мові Python:

1. Topo: Це базовий клас для топологій Mininet;

2. Add Host (name, cpu = f): Він використовується для додавання хоста до топології, яка містить два параметри. Перший параметр вказує ім'я хоста, а другий - частку загальних системних ресурсів ЦП, які мають бути виділений віртуальному хосту;

3. Add Switch(): Він використовується для додавання комутатора до топологію і повертає ім'я комутатора, наприклад s1;

4. Add Link (node1, node2, ** link options): Використовується для додавання двонаправленого посилання, яке містить три параметри. Перший, другий параметр вказують ім'я хоста та комутатора відповідно, а третій параметр визначає словник, що містить кількість параметрів, таких як пропускна здатність, характеристики затримки та втрати, з максимальним розміром черги;

5. start(): Використовується для запуску мережі;

6. stop(): Використовується для зупинки мережі;

7. Mininet: Він використовується як основний клас для створення та керувати мережею;

8. net. hosts: Використовується для показу всіх хостів у мережі;

9. ping All (): Це використовується для перевірки підключення між усіма вузлами;

10. Set Log Level: існує декілька рівнів, такі як інформація, налагодження та вихід;

11. Dump Node Connections (): підключення до/з набору вузлів.

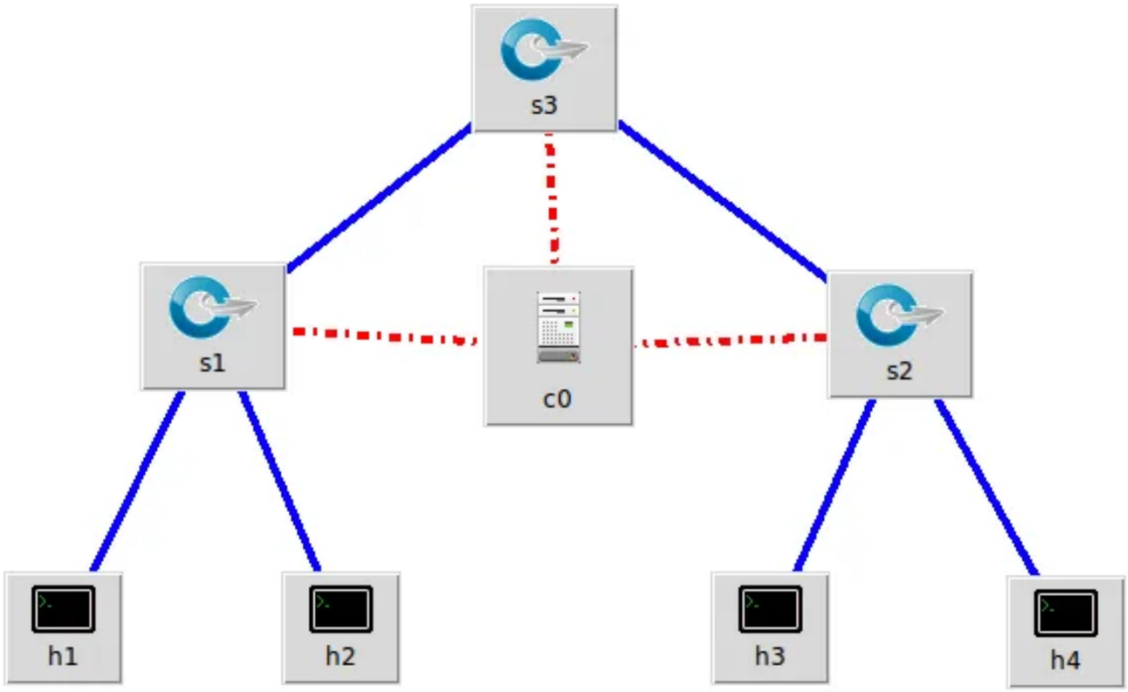


Рис. 3.7 – Побудована топологія мережі на основі SDN-мереж

Контролер POX був запущений з додатком Simple_Switch. Додаток комутатора відстежує, де знаходиться хост з MAC-адресою, і відповідно відправляє пакети до пункту призначення, а не заповнює його через усі порти. Перемикач SDN (наприклад, перемикач OpenFlow), контролер SDN та інтерфейси, присутні на контролері для зв'язку з пристроями переадресації, як правило, інтерфейс на південь (OpenFlow) та інтерфейс мережевих додатків (інтерфейс на північ) є основними будівельними блоками Розгортання SDN. Комутатори в SDN часто представлені як базове обладнання для переадресації, доступне через відкритий інтерфейс, оскільки логіка управління та алгоритми завантажуються на контролер. Комутатори OpenFlow бувають двох різновидів: чисті (лише для OpenFlow) та гібридні (з підтримкою OpenFlow).

Комутатор OpenFlow - це основний елемент переадресації, який доступний через протокол та інтерфейс OpenFlow. Хоча на перший погляд це налаштування може спростити апаратне переключення, архітектури SDN на основі потоку, такі як OpenFlow, можуть вимагати додаткових записів таблиці переадресації, буферного простору та статистичних лічильників. У мережі OpenFlow комутатори мають два варіанти: гібридний (з підтримкою OpenFlow) і чистий (лише OpenFlow).

Гібридні комутатори підтримують OpenFlow на додаток до традиційних операцій та протоколів (комутація L2/L3). Комутатор OpenFlow складається з таблиці потоків, яка виконує пошук та пересилання пакетів. Кожна таблиця потоків у комутаторі містить набір записів потоку, який складається з:

- Поля заголовка або поля відповідності з інформацією, знайденою в заголовку пакета, вхідному порту та метаданих для відповідності вхідних пакетів.
- Лічильників, що використовуються для збору статистичних даних для конкретного потоку, таких як кількість отриманих пакетів, кількість байт і тривалість потоку.
- Набору інструкцій або дій, що застосовуються після відповідності, яка диктує, як обробляти відповідні пакети. Наприклад, дією може бути переадресація пакету на вказаний порт.

  Логіка обробки пакетів OpenFlow Switch:

```
Create MAC table
if (packet into switch)
{ parse packet to reveal src and dst MAC addr
store in dictionary mapping between MAC and portl
lookup dst MAC into port dictionary of switch to find next hop if (next
hop is found) {
create flow mode
send
}
else {
flood all ports=in_port
}
```

Запуск mininet з трьома хостами та одним комутаторів OpenFlow s1:

```
mininet@mininet-vm:~$ sudo mn --controller=remote,192.168.0.12 --mac --
topo=single,2 --switch=ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s2) (h2, s2), (h3, s3), (h4, s3), (s1, s2), (s3, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2...
*** Starting CLI:
mininet&gt; h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.746 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.365 ms
^C
--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.365/0.555/0.746/0.191 ms
```

Потоки, створені в Mininet:

```
mininet@mininet-vm:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
cookie=0x2b00000000000011, duration=401.344s, table=0, n_packets=7,
n_bytes=518, priority=2,in_port=1 actions=output:2,CONTROLLER:65535
cookie=0x2b00000000000010, duration=401.344s, table=0, n_packets=6,
n_bytes=476, priority=2,in_port=2 actions=output:1,CONTROLLER:65535
cookie=0x2b0000000000000b, duration=405.347s, table=0, n_packets=0, n_bytes=0,
priority=100,dl_type=0x88cc actions=CONTROLLER:65535
cookie=0x2b0000000000000b, duration=405.34s, table=0, n_packets=0, n_bytes=0,
priority=0 actions=drop
```

3.3. Тестування додатку

В експерименті була використана топологія Mininet, подана на малюнку 3.6. Ця топологія включає три комутатори OpenFlow, підключені до чотирьох хостів, і один контролер OpenFlow. Після запуску середовища емуляції Mininet з цією топологією контролер і комутатор OpenFlow ініціюють емуляцію SDN-мережі з протоколом OpenFlow, яку можна перехопити та переглянути у Wireshark, яке показано на малюнку 3.8. На основі цієї емульованої платформи SDN, POX використовується як контролер SDN. На наступному скріншоті показано захоплений трафік, який показує повідомлення Hello, запит/відповідь. Це підтверджує, що комутатор OpenFlow у цій установці підключений до контролера OpenFlow. Тепер ми можемо перевірити підключення кожного хоста за допомогою простої команди:

ping: mininet> h1 ping –c 3 h2

а результат запиту команди ping продемонстровано на рисунку 3.8, а на рис. 3.9 показаний захоплений пакет.

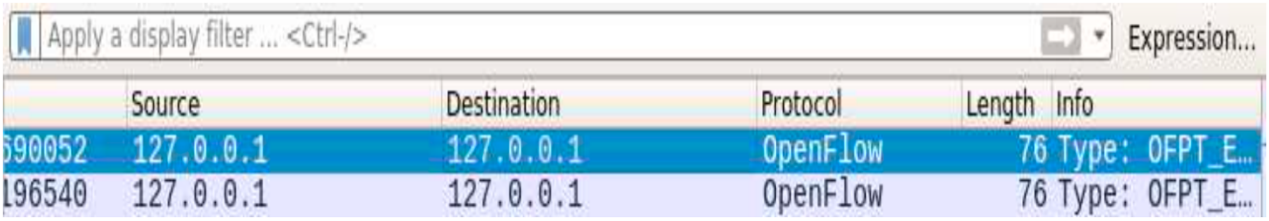
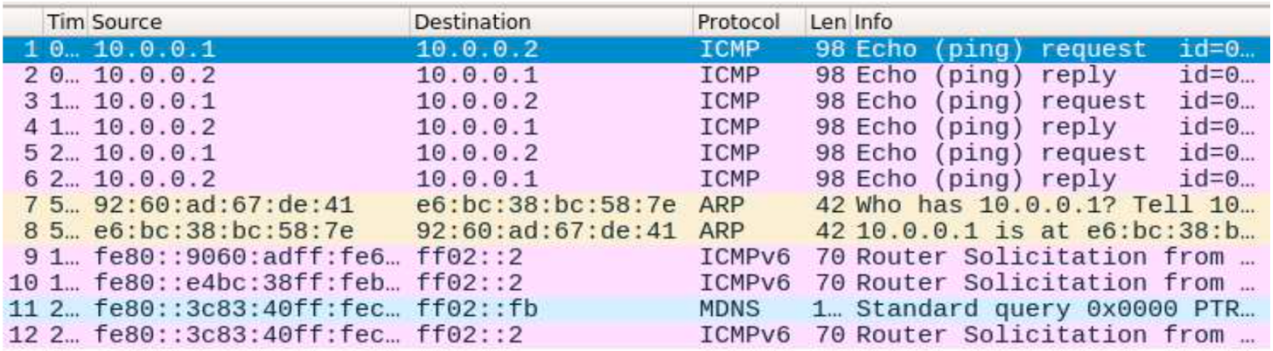


Рис. 3.8 – OpenFlow трафік, відстежений у Wireshark



	Tim	Source	Destination	Protocol	Len	Info
1	0…	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0…
2	0…	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0…
3	1…	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0…
4	1…	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0…
5	2…	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0…
6	2…	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0…
7	5…	92:60:ad:67:de:41	e6:bc:38:bc:58:7e	ARP	42	Who has 10.0.0.1? Tell 10…
8	5…	e6:bc:38:bc:58:7e	92:60:ad:67:de:41	ARP	42	10.0.0.1 is at e6:bc:38:b…
9	1…	fe80::9060:adff:fe6…	ff02::2	ICMPv6	70	Router Solicitation from …
10	1…	fe80::e4bc:38ff:feb…	ff02::2	ICMPv6	70	Router Solicitation from …
11	2…	fe80::3c83:40ff:fec…	ff02::fb	MDNS	1…	Standard query 0x0000 PTR…
12	2…	fe80::3c83:40ff:fec…	ff02::2	ICMPv6	70	Router Solicitation from …

Рис. 3.9 – Захоплений трафік після виконання команди h1 ping –c 3 h2 у Mininet

Пінг відправляє три пакети запитів пінгу, як показано на малюнку 3.9. Запис потоку, що охоплює пінг-трафік ICMP, раніше був встановлений у комутаторі, тому не створювався контрольний трафік, і пакети негайно проходять через комутатор. Більш простий спосіб запустити цей тест - скористатися вбудованою командою pinall в Mininet CLI, яка виконує пінг усіх пар. Ще одним корисним тестом є автономний регресійний тест. Наступна команда створила мінімальну топологію, запустила контрольний контролер OpenFlow, запустила тест перевірки пінгу всіх пар і зруйнувала як топологію, так і контролер. Швидкість передачі даних кожного емульованого Ethernet-зв'язку в Mininet забезпечується Linux Traffic Control, який має ряд планувальників пакетів для формування трафіку до заданої швидкості. Mininet дозволяє встановити параметри зв'язку, і їх навіть можна встановити автоматично з командного рядка:

```
sudo mn –link tc, bw=10, delay=10ms
```

Перевірити пропускну здатність трафіку за допомогою команди та результат:

```
mininet> iperf
*** Iperf: testing TCP bandwith between h1 and h2
*** Results: [‘9.50 Mbit/sec’, ‘11.9 Mbits/sec’]
```

Це встановить пропускну здатність посилань 10 Мбіт/с та затримку 10 мс. Враховуючи це значення затримки, час зворотного зв'язку повинен становити близько 40 мс, оскільки запит ICMP проходить два посилання (одне до комутатора, одне до пункту призначення), а відповідь ICMP проходить два посилання, що повертаються. Час туди-назад для пакету проілюстровано на рис.. Як показано в Таблиці 1, час у зворотному напрямку є більшим у лінійній топології порівняно з трійниковою та гібридною топологією.

Нижче показано час для кожного захопленого пакету:

```
mininet> h1 ping -c 10 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=46.4 ms
```

Висновки до розділу 3

У розділі наведені результати розробки та тестування програмного продукту, а точніше мережевого рішення у вигляді побудованої топології, яка була створена та налаштована у середовищі емуляції Mininet, логіка передачі пакетів реалізована на Python.

У Mininet була інтегрована концепція SDN-мереж за допомогою використання правил протоколу OpenFlow в обміні даними між мережевими елементами. Для підтвердження коректної роботи програми, було досліджено та проаналізовано трафік у мережі за допомогою додатку Wireshark.

ВИСНОВКИ

Сфера IoT сьогодні є однією з найбільш швидкозростаючих, і нові пристрої IoT впроваджуються щодня. Завдяки такому динамічному типу середовища традиційні мережі не є найкращим варіантом для задоволення вимог IoT. Існує потреба в більш динамічній та безпечній мережевій інфраструктурі для підтримки операцій IoT. Як було сказано раніше, SDN - це нова технологія, яка дозволяє повністю контролювати загальну поведінку мережі та запобігає перевантаженню мережі. Крім того, контролер SDN надає корисні інструменти налагодження, які можуть використовуватися середовищем IoT для підвищення безпеки.

Програмно-конфігурована мережа (SDN), яку часто позначають як революційну ідею в комп'ютерних мережах, обіцяє значно спростити мережевий контроль, управління та забезпечити інновації завдяки програмованості мережі, що дозволить сфері IoT вирішити одну з головних проблем – безпека та конфіденційність. Mininet сприяє створенню та керуванню програмно-конфігурованими мережевими компонентами. Mininet - це платформа для швидкого прототипування мережі. Це альтернатива проведенню експериментів SDN в емульованих мережах. Mininet корисний для вивчення OpenFlow, який є відкритим інтерфейсом для управління мережевими елементами через їх таблиці переадресації. Мережевий елемент можна перетворити на комутатор або маршрутизатор за допомогою примітивів низького рівня, визначених у OpenFlow.

У цьому дослідженні була імітована програмно-конфігурована мережа, використовуючи Mininet і POX, алгоритм реалізації передачі пакетів, розроблений на Python, яка була використана як контролер з комутатором OpenFlow, обговорені характеристики програмно-конфігурованої мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S. Haller, S. Karnouskos, and C. Schroth, "The Internet of Things in an Enterprise Context," in Future Internet – FIS 2008 Lecture Notes in Computer Science Vol. 5468, 2009, pp 14-28.
2. A. C. Sarma, and J. Girão, "Identities in the Future Internet of Things," in Wireless Personal Communications 49.3, 2009, pp. 353-363.
3. Sibylle Schallera and Dave Hoodb, “Software defined networking architecture standardization,” Computer Standards & Interfaces, Vol. 54, pp. 197–202, 2017.
4. Chan, M., Campo, E., Estève, D., & Fourniols, J. Y. (2009). Smart homes - current features and future perspectives. Maturitas, 64(2), 90
5. Fang, X., Misra, S., Xue, G., & Yang, D. (2012). Smart grid — the new and improved power grid: a survey. IEEE Communications Surveys & Tutorials, 14(4), 944-980.
6. Kaneko, M., Arima, K., Murakami, T., Isshiki, M., & Sugimura, H. (2017). Design and implementation of interactive control system for smart houses. IEEE International Conference on Consumer Electronics (pp.283-284). IEEE.
7. Palm, J. (2009). Emergency management in the swedish electricity grid from a household perspective. Journal of Contingencies & Crisis Management, 17(1), 55–63.
8. W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, “A survey on softeware-define networking,” IEEE Commun Surveyw & Tutorials, vol. 17, no. 1, pp. 27-51, 2015.
9. I. F. Akyildiz, A. Lee, P. Wang, M. Luo and W. Chou, "A roadmap for traffic engineering in SDN- OpenFlow networks," Computer Networks, vol. 71, no. 1, pp. 1-30, 2014.
10. OpenFlow Switch Specifcation - Version 1.3.5., March 2015. [Online].
11. P. Ben and D. Bruce, “The Open vSwitch Database Management Protocol. RFC 7047,” 2013.

12. F. Callegati, W. Cerroni, C. Contoli and G. Santandrea, "Implementing dynamic chaining of Virtual Network Functions in OpenStack platform," 17th International Conference on Transparent Optical Networks (ICTON), 2015.
13. A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, R. G. L. Dong and J. Wang, "Forwarding and Control Element Separation (ForCES) Protocol Specification," Internet Engineering Task Force (IETF), 2010.
14. E. Haleplidis, D. Joachimpillai, J. H. Salim, D. Lopez, J. Martin, K. Pentikousis, S. Denazis and O. Koufopavlou, "ForCES Applicability to SDN-Enhanced NFV," Third European Workshop on Software Defined Networks, pp. 43-48, 2014.
15. E. Haleplidis, J. H. Salim, S. Denazis and O. Koufopavlou, "Towards a Network Abstraction Model for SDN," Journal of Network and Systems Management, vol. 23, no. 2, pp. 309-327, 2014.
16. P. Saint-Andre, XMPP The Defnite Guide, Safari Book, O'Reilly, 2009.
17. Cisco OpFlex, , "Cisco Application Centric Infrastructure Policy-Based Redirect Service Graph Design," [Online]. Available: https://www.cisco.com/c/en/us/solutions/data-center virtualization/application- centric-infrastructure/white-paper-c11-739971.html.
18. R. T. Fielding, "Chapter 5: representational state transfer (REST)," Architectural Styles and the Design of Network-Based Sofware Architectures, pp. University of California, Irvine, Calif USA, 2000.
19. Build SDN Agilely, "COMPONENT-BASED SOFTWARE DEFINED NETWORKING FRAMEWORK," [Online]. Available: http://osrg.github.com/ryu/.
20. OpenDayLight, "http://www.OpenDayLight.org/project/technical-overview".
21. L. Li, W. Chou, W. Zhou and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," IEEE Transactions on Network and Service Management, vol. 13, no. 1, pp. 154-167, 2016.

22. A. Voellmy, H. Kim and N. Feamster, “Procera: A Language for High-level Reactive Network Control,” pp. 43-48, 2012.
23. N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story and D. Walker, "Frenetic: A Network Programming Language," 16th ACM SIGPLAN International Conference on Functional Programming (ICFP ’11), p. 279–291, 2011.
24. H. Cummins and T. Ward, Enterprise OSGi in Action, Manning 1st edition, 2013.
25. D. Erickson, “The beacon openflow controller,” 2nd ACM SIGCOMM Workshop on Hot Topics in Sofware Defned Networking (HotSDN ’13), pp. 13-18, 2013.
26. Internet Engineering Task Force (IETF) , " YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," https://tools.ietf.org/html/rfc6020, no. RFC 6020, 2010.
27. R. Hirannaiah, "Overview of Model-Driven SAL and Creating and Application based on MD-SAL," http://events.linuxfoundation.org/sites/events/fles/slides/Radhika Hirannaiah MD-SAL ONS2016.pdf, 2016.
28. D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Sofware- defined networking: a comprehensive survey," IEEE Proceedings, vol. 103, no. 1, pp. 14-76, 2015.
29. M. Pham and D. B. Hoang, "SDN applications-the intent based Northbound Interface realisation for extended applications," IEEE NetSof Conference and Workshops (NetSof ’16), p. 372–377, 2016.
30. Cisco, “Microservices Infrastructure,” https://github.com/CiscoCloud/microservices-infrastructure.
31. Open Network Foundation, "The ONOS Project," http://onosproject.org/, 2017.

32. ONF, "Blog: Intent, what. Not how," https://www.opennetworking.org/?p=1633&option=com wordpress&Itemid=155.
33. ODL, "Network Modelling Language (NEMO)," https://wiki.OpenDayLight.org/view/Network Intent Composition:NEMO_Model.
34. B. J. v. Asten, N. L. M. v. Adrichem and F. A. Kuipers, "Scalability and resilience of sofware-defned networking: an overview," https://arxiv.org/abs/1408.6760, 2014.
35. Nobakht, M.; Sivaraman, V.; Boreli, R. A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT using OpenFlow. In Proceedings of the IEEE 11th International. Conference Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 147–156.
36. Bull, P.; Austin, R.; Popov, E.; Sharma, M.; Watson, R. Flow Based Security for IoT Devices Using an SDN Gateway. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016; IEEE: 2016; pp. 157–163.
37. Kuipers, F.A. SDN and virtualization solutions for the Internet of Things: A survey. IEEE Access 2016, 4, 5591–5606.
38. S. Scott-Hayward, G. OCallaghan, and S. Sezer, “SSDN security: A survey”, in Proceedings of the IEEE SDN for Future Networks and Services. pp.1-7, 2013.
39. S. Son, S. Shin, V. Yegneswaran, P. Porras, and G.Gu, “Model checking invariant security properties in openflow”, in Proceedings of the IEEE International Conference on Communications. pp. 1974-1979, 2013.
40. H. Hu, W. Han, G.J. Ahn, and Z. Zhao, “Flowguard: Building robust firewalls for software-defined networks”, in Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. pp. 97–102, 2014.

41. R. Skowyra, S. Bahargam, and A. Bestavros, “Software-defined ids for securing embedded mobile devices”, in Proceedings of the Workshop of Research and Educational. pp. 84-48, 2014.
42. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” SIGCOMM Computer Commununication.
43. M. Mendonc ̧a, B. N. Astuto, X. N. Nguyen, K. Obraczka, and T. Turletti, “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks,” IEEE 37th Conference on. IEEE, 2013, pp. 311–315.
44. Faris Keti and Shavan Askar. “Emulation of Software Defined Networks Using Mininet in Different Simulation Environments,” 6th International Conference on Intelligent Systems, Modeling and Simulation, 2015.
45. Danda B. Rawat, “Software Defined Networking Architecture, Security and Energy Efficiency: A Survey,” IEEE Communication Surveys & Tutorials, Vol. 19, No. 1, 2017.
46. Diego Kreutz, “Software-Defined Networking: A Comprehensive Survey,” IEEE, Vol. 103, No. 1, 2014.
47. K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, “Software-defined networking (SDN): A survey,” Secur. Commun. Netw., Vol. 9, No. 18, 2017.
48. R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOXIOpenFlow, in Local Computer Networks (LCN)”,2010 IEEE 35th Conference on. IEEE, 2010, pp. 408–415.
49. H. Jafarian, E. Al-Shaer, and Q. Duan, “Open flow random host mutation: transparent moving target defense using software defined networking”, in Proceedings of the first workshop on Hot topics in software defined networks, ACM, 2012, pp. 127–132.

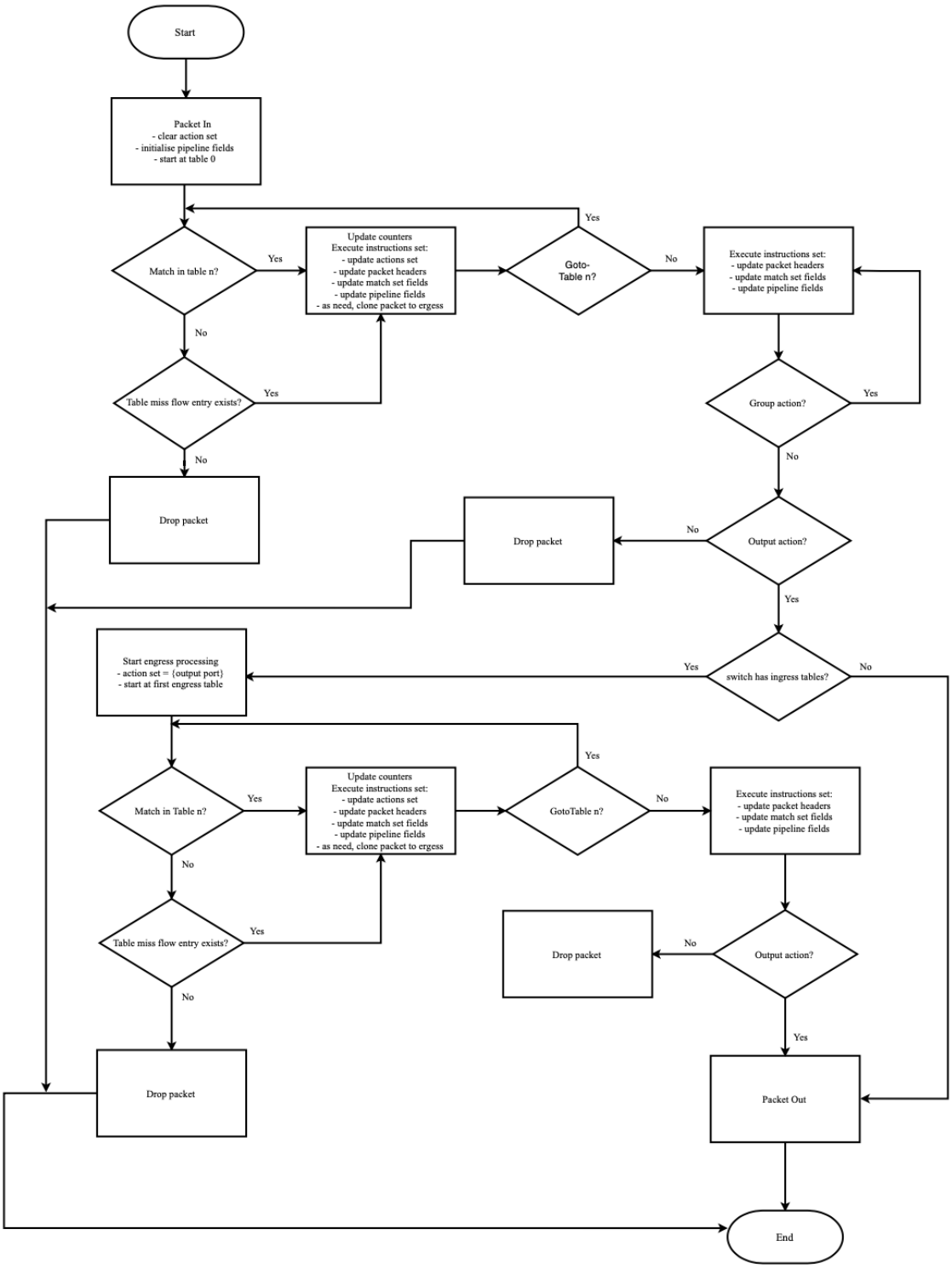
ДОДАТОК 1

Спосіб організації топології для системи «розумного будинку»
на основі SDN-мереж

СХЕМА АЛГОРИТМУ

Аркушів 1

Київ – 2021



					ІАЛЦ.467200.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Овчарова Ю. В.			*Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж. Схема алгоритму*	Лит. / Арк 1 / Аркушів 1
Перев.		Алещенко О. В.				
Реценз.						НТУУ «КПІ», ФІОТ, ІВ-72
Н. Контр.		Сімоненко В. П.				
Затверд.		Стіренко С. Г.				

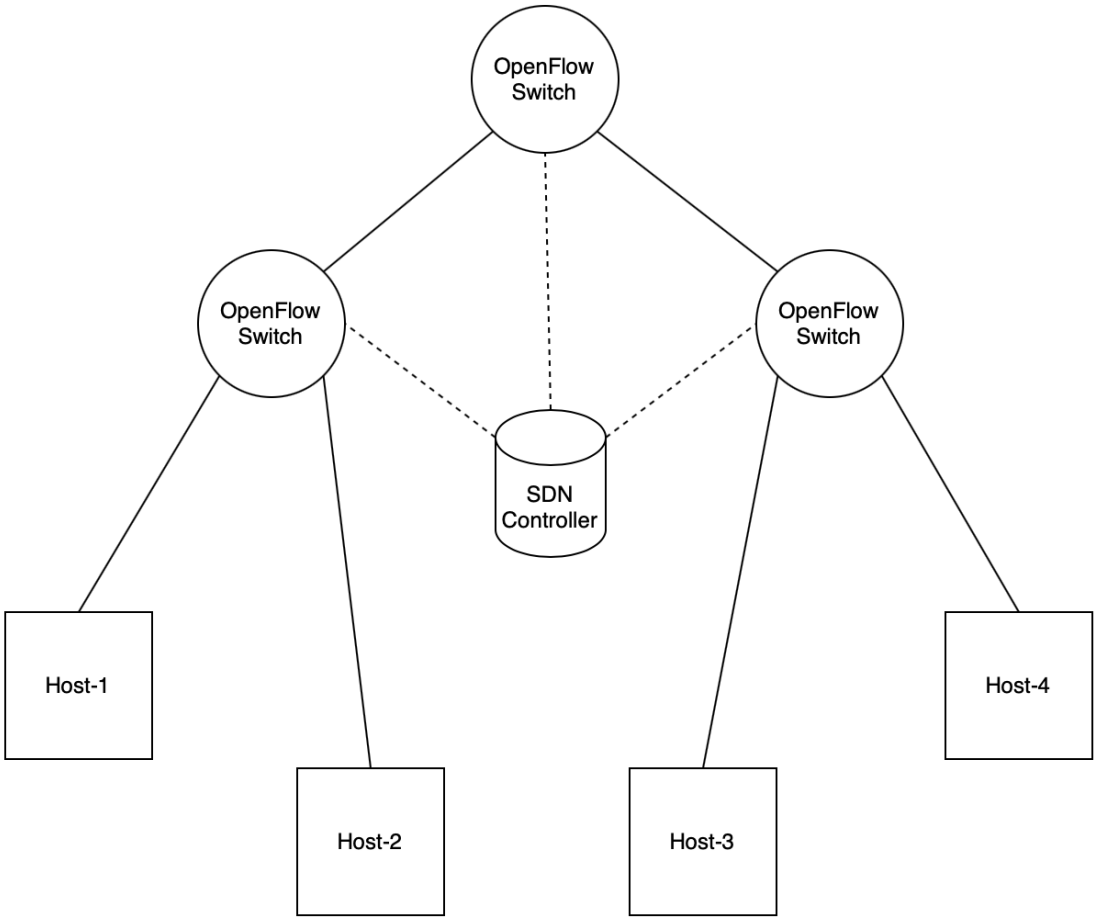
ДОДАТОК 2

Спосіб організації топології для системи «розумного будинку»
на основі SDN-мереж

СТРУКТУРНА СХЕМА

Аркушів 1

Київ – 2021



					ІАЛЦ.467200.004 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Овчарова Ю. В.			*Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж. Структурна схема*	Лит. / Арк. 1 / Аркушів 1
Перев.		Алещенко О. В.				
Реценз.						НТУУ «КПІ», ФІОТ, ІВ-72
Н. Контр.		Сімоненко В. П.				
Затверд.		Стіренко С. Г.				

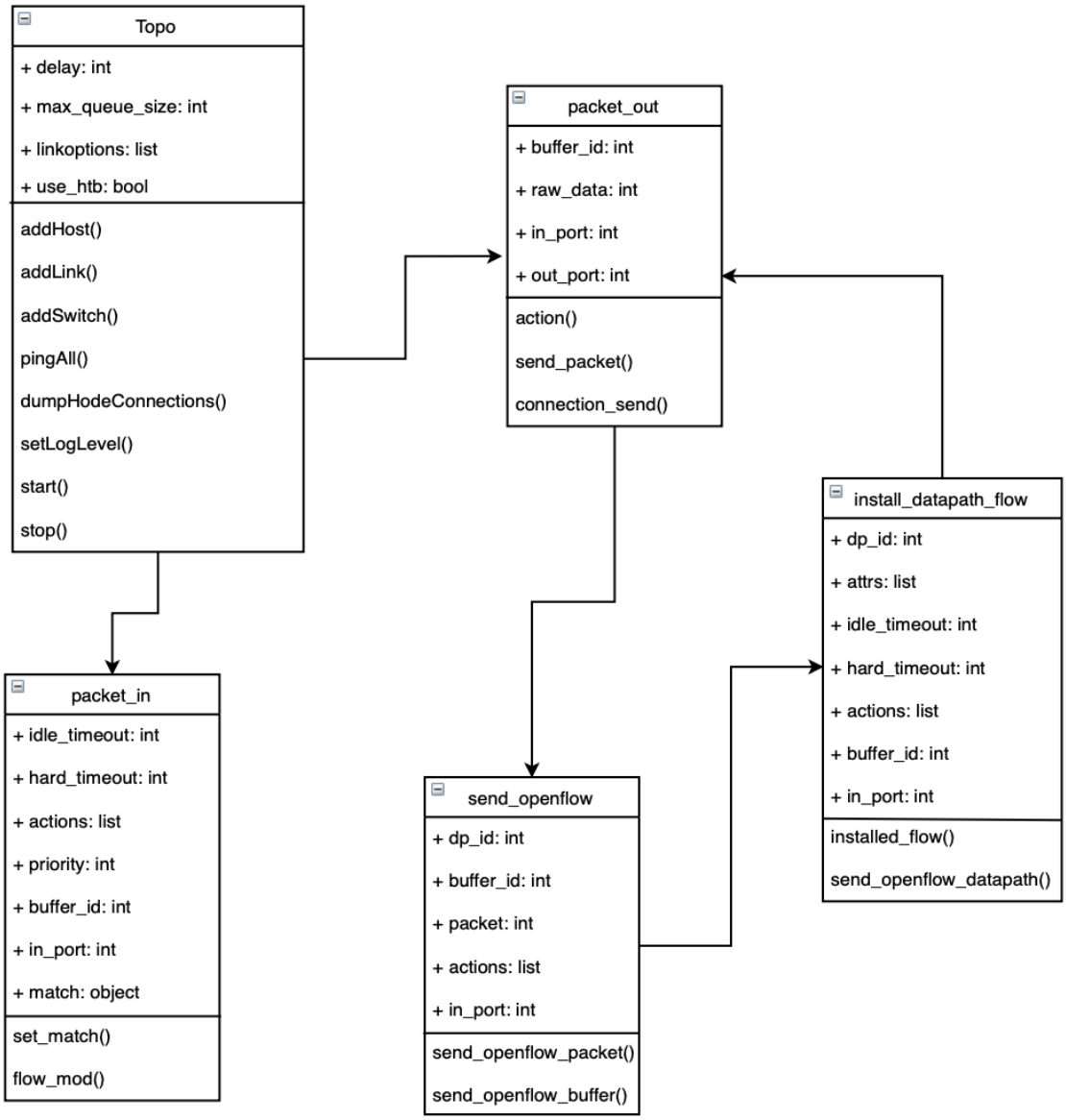
ДОДАТОК 3

Спосіб організації топології для системи «розумного будинку»
на основі SDN-мереж

ДІАГРАМА КЛАСІВ

Аркушів 1

Київ – 2021



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.004 Д3		
Розроб.		Овчарова Ю. В.			*Спосіб організації топології для системи «розумного будинку» на основі SDN-мереж. Діаграма класів*	Лит.	Арк. 1 / Аркушів 1
Перев.		Алещенко О. В.					
Реценз.						НТУУ «КПІ», ФІОТ, ІВ-72	
Н. Контр.		Сімоненко В. П.					
Затверд.		Стіренко С. Г.					