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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
“___”_____20__ р

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп’ютерна інженерія»

на тему: «Удосконалення мереж транспортних засобів за допомогою технології SDN»

Виконав:
студент _IV_курсу, групи__ІО-61__

___Косюга Олексій Євгенійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _асистент Калюжний Олександр Олегович__ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант __н. контроль_ _доц. д.т.н. Сімоненко В.П._ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент ____доцент, к.т.н. Орлова Марія Миколаївна____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
"___"_____20__ р.

ЗАВДАННЯ

на дипломний проект студенту

Косюзі Олексію Євгенійовичу

1. Тема проекту «*Удосконалення мереж транспортних засобів за допомогою технології SDN»*

керівник проекту Калюжний Олександр Олегович, асистент, затверджені наказом по університету від *«__07__»* *__травня__* 2020р. №__1081-с__

2. Термін здачі студентом закінченої роботи_____*2020р.*

3. Вихідні дані до проекту *технічне завдання, теоретичні дані.*

4. Зміст пояснювальної записки: *опис предметної області, опис проблеми, існуючих рішень, формулювання власного рішення, вибір та порівняння засобів для написання додатку, опис архітектури та графічного інтерфейсу додатку.*

5. Консультант роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

6. Дата видачі завдання *01.09 .2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	*Затвердження теми роботи*	01.09.2019	
2.	*Вивчення та аналіз завдання*	15.09.2019	
3.	*Розробка архітектури та загальної структури систем*	04.10.2019	
4.	*Розробка структур окремих підсистем*	13.12.2019	
5.	*Програмна реалізація системи*	03.02.2020	
6.	*Оформлення пояснювальної записки*	04.04.2020	
7.	*Передзахист*	26.05.2020	
8.	*Захист*	25.06.2020	

Студент Олексій КОСЮГА _____

(підпис)

Керівник Олександр КАЛЮЖНИЙ _____

(підпис)

Анотація

В бакалаврському дипломному проекті розроблено архітектуру SDN VANET та програмний додаток для захисту її від DDOS атак. Запропоноване рішення дозволяє автоматично знаходити та фільтрувати потенційних зловмисників та мануально налаштовувати рівень доступу клієнтів у мережі. Продукт створений на основі Floodlight контролера на мові Python, моделювання мережі відбувається за допомогою Mininet-WiFi симулятора.

Annotation

The bachelor's thesis project developed the SDN VANET architecture and a software application to protect it from DDOS attacks. The proposed solution allows you to automatically find and filter potential attackers and manually adjust the level of customer access to the network. The product is based on the Floodlight controller in Python, the network is simulated using a Mininet-WiFi simulator.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.045430.001 ВП	Відомість проекту	1	
3	А4	ІАЛЦ.045430.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.045430.003 ПЗ	Пояснювальна записка	64	
5	А3	ІАЛЦ.045430.004 Д1	Принципова схема класів брандмауера	1	
6	А3	ІАЛЦ.045430.005 Д2	Структурна схема мережі SDN VANET з брандмауером	1	
7	А3	ІАЛЦ.045430.006 Д3	Функціональна схема алгоритму роботи брандмауера	1	
8	А4	ІАЛЦ.045430.007 ДА	Лістинг системи	14	

					ІАЛЦ.045430.001 ВП	
Зм.		*№ документа*	*Підпис*	*Дата*		
Розробив		*Косюга О.Є.*			Відомість дипломного проекту	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		Калюжний О.О.				*1* / *1*
Н. Контр.		*Сімоненко В.П.*				*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр. ІО-61*
Затвердив		*Стіренко* С.Г.				

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня___бакалавр___

на тему: “ Удосконалення мереж транспортних засобів за допомогою технології SDN ”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку курсу «Інженерія програмного забезпечення». Область застосування: практичне використання людьми при користуванні комп’ютерними засобами.

2. ПІДСТАВИ ДЛЯ 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 ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка архітектури SDN VANET та програмного додатку для її захисту від DDOS атак.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування, бакалаврські роботи інших студентів, публікації в Інтернеті з даних питань.

5. **ТЕХНІЧНІ ВИМОГИ**

5.1. Вимоги до розробляємого продукту

- Розроблена архітектура повинна бути створена згідно стандартам SDN
- Програмний додаток повинен автоматично блокувати потенційних зловмисників.
- Брандмауер повинен забезпечити адміністратора можливість переглядати лог-файли мережі та мануально обмежувати доступ клієнтів.

5.2. Вимоги до програмного забезпечення

- Linux-подібна операційна система.
- Наявність на комп’ютері Python не нижче версії 2.7.

5.3. Вимоги до апаратної частини

- Оперативної пам’яті не менше 512 Мбайт.
- Вільне місце на жорсткому диску не менше 500 Мбайт.
- Процессор 1.4 ГГц

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	28.03.2020
Складання і узгодження технічного завдання	03.04.2020
Створення модулів системи, що розробляється	15.04.2020
Тестування окремих модулів системи	25.04.2020
Допрацювання, налагодження і виправлення помилок	01.05.2020
Оформлення документації дипломної роботи	15.05.2020

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту
на тему: «Удосконалення мереж транспортних засобів за допомогою технології SDN»

ЗМІСТ

ВСТУП

Актуальність дослідження. З урахуванням масштабу розвитку засобів передачі інформації, на сьогоднішній день важко недооцінювати важливість як подальшого вдосконалення вже існуючих рішень, так і впровадження інноваційних методів і технологій. Таким чином, подальший розвиток перспективних напрямів, таких як бездротові мобільні мережі, що вже інтегрувалися у повсякденне життя як будь-якого користувача мобільного телефону, так і використовуються для вирішення низки конкретних бізнес-задач різного плану, є пріоритетною задачею.

Однією з найбільш важливих сфер подальшого розвитку даного напряму є створення спеціальних бездротових мереж для забезпечення потреб водіїв транспортних засобів. Використання таких мереж дозволить суттєво покращити умови дорожнього руху через підтримку інформованості його учасників про будь-які зміни, впровадження способів дистанційного аналізу його стану при несприятливих умовах, налагодження зв'язку зі спеціальними програмними додатками для комфорту водіїв під час руху.

Проблемою створення таких мереж є природні особливості поведінки учасників дорожнього руху та неефективність використання традиційний підходів до налаштування мережевою архітектури. Це пояснюється динамічністю описуваної системи, великою кількістю учасників та масштабністю покриття мережі для її ефективного використання. Через це дані мережі створюються згідно принципу інтеграції мережевої архітектури безпосередньо у кожен вузол системи, що водночас стають її клієнтами та безпосередніми учасниками пересилки даних. Мережі транспортних засобів, побудовані згідно даного принципу, називаються VANET мережами.

Незважаючи на те, що на даний момент існує велика кількість потенційно ефективних реалізацій даного типу мереж, повсякденне використання такої технології пов'язане з вирішенням переліку завдань по їх

оптимізації та покращення задля забезпечення їх економічної доцільності. Головним проблемами, що виникають у процесі реалізації, є контроль мережі, забезпечення безпеки її учасників та безпосередньо мережі, підбір ефективних протоколів маршрутизації або ж винайдення нових.

Пошук шляхів подолання цих перешкод є наступним кроком у інтеграції VANET мереж для повсякденного використання та суттєвого покращення стану дорожнього руху і усіх його учасників.

Одним з перспективних напрямів є використання сучасних концептів мережевих технологій, таких як SDN, для вирішення вже визначених проблем VANET мереж, таких як безпека, ефективне використання ресурсів мережі, маршрутизація та впровадження інструментів для її моніторингу та контролю.

Предметом дослідження даної роботи є VANET мережа з інтеграцією SDN.

Метою даної роботи є удосконалення безпеки та ефективності мереж транспортних засобів на основі інтеграції технології SDN. Для досягнення даної мети було поставлено та вирішено такі завдання:

1. Розглянути існуючі варіанти VANET мереж, проаналізувати їх особливості, недоліки та переваги.
2. Обґрунтувати необхідність створення VANET мережі з інтеграцією SDN технології, огляд існуючих рішень.
3. Розробка архітектури VANET.
4. Реалізація механізму SDN у мережі VANET, розробка симуляції мережі.
5. Експериментальне випробування мережі та аналіз отриманих даних.

.

РОЗДІЛ 1
АНАЛІЗ КЛАСИЧНИХ VANET МЕРЕЖ

1.1. Огляд концепту VANET мереж

VANET (Vehicular ad-hoc networks) — тип децентралізованих бездротових мереж транспортних засобів. Концепт VANET мереж був створений як конкретна імплементація архітектури мереж MANET (Mobile ad hoc network) для створення мереж транспортних засобів, а отже — наслідує їх основні особливості, переваги та недоліки. Вузлами такої мережі є транспортні засоби, що постійно змінюють своє місцеположення відносно один одного, та елементи дорожньої інфраструктури, які зазвичай визначаються як RSU (Road-Side Unit, придорожній мережевий вузол), що забезпечують статичні точки зв'язку у мережі та надання доступу транспортним засобам до програмних додатків різних типів [1].

Дані мережі будуються згідно принципу ad hoc — кожен її вузол водночас є користувачем та безпосереднім учасником процесу роутингу і відповідної передачі даних. Зв'язок між ними будується на основі DSRC (Dedicated short-range communications) — односторонньому або двосторонньому радіозв'язку близької дії.

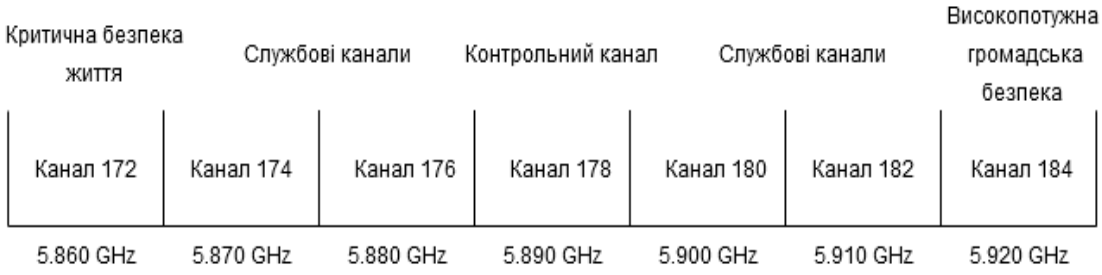


Рис.1.1. Діаграма каналів у DSRC

На основі даної комунікації створюється зв'язок типу V2V (Vehicle to Vehicle, ТЗ до ТЗ) або зв'язок з дорожньою інфраструктурою типу V2I (Vehicle to Infrastructure).

Класичний варіант даних мереж є децентралізованим, внутрішня інфраструктура мережі підтримується кожним з вузлів – безпосередніми транспортними засобами, та придорожніми блоками зв’язку. Дані блоки можуть виконувати функції маршрутизаторів, мостів, точок доступу та безпосередніх клієнтів мережі.

Таким чином, з точки зору структурного рівня VANET мережі є самоорганізованими з динамічною топологією: зв’язки між вузлами мережі встановлюються та втрачаються хаотично, у відповідності до сили розповсюдження сигналу та віддаленості між транспортними засобами. Це унеможливлює використання статичної топології для моделювання мережі.

Передача даних та контроль трафіку відбувається згідно визначеного алгоритму маршрутизації, пошук якого зазвичай є однією з найважливіших задач для ефективної реалізації мережі внаслідок складностей, викликаних її топологією.

Визначення ефективної політики безпеки мережі та відповідних засобів є ще одним складним питанням при реалізації даного типу мереж, адже особливість їх структури, децентралізованість та висока динамічність створення та втрати зв’язків суттєво ускладнюють задачу забезпечення гідного рівня безпеки даних користувачів та захисту мережі від класичних та специфічних типів атак.

Створення ефективних та доступних VANET мереж дозволить вирішити наступні питання з приводу оптимізації дорожнього руху:

- підвищення рівня безпеки дорожнього руху за рахунок постійного та детального оповіщення водія про можливі небезпечні умови на дорозі та загальний стан дорожнього руху згідно його маршруту;
- реєстрація місцеположення супутніх транспортних засобів у мережі з різною метою: запобігання ДТП у місцях підвищеної небезпеки, визначення пробок, тощо;

- використання інформації про стан дорожнього руху для побудови оптимального шляху — особливо важливо для спецслужб, швидкої, тощо;
- оповіщення водіїв про наявність ТЗ підвищеного пріоритету на дорозі;
- подальше спрощення певних аспектів дорожнього руху, таких як сплачування податків за використання платних доріг, парковку;
- збір та аналіз інформації про користування тим чи іншим дорожнім шляхом та його учасників;
- дистанційна перевірка безпеки ТЗ і т.д.

Таким чином, коректне імплементування VANET мереж для побутового використання може суттєво покращити стан дорожнього руху; зменшити кількість аварійних випадків та ДТП; забезпечити підвищений рівень комфорту водіїв; забезпечувати їх повним набором інформації про дорожній рух незалежно від інших зовнішніх умов, таких як несприятливі погодні умови; проводити повний динамічний аналіз дорожнього трафіку. Потенційно, використання даної технології може бути використано для максимальної автоматизації процесу дорожнього руху та або імплементації повністю автономних транспортних засобів, що суттєво зменшить ризики нанесення шкоди пасажирам або товару та стане відповідним наступним шляхом у розвитку транспортування [2].

1.2. Аналіз структури VANET мереж

Згідно визначення, типова VANET мережа включає у собі певну кількість мобільних вузлів — транспортних засобів, — та дорожньої інфраструктури у виді RSU для забезпечення зв’язку. Таким чином, усі ноди мережі можна поділити на 2 окремих типа. Головними особливостями структури є розподіл внутрішньої архітектури по вузлам зв’язку та відсутність центральних вузлів управління. Макет типової архітектури VANET мережі представлено на рисунку 1.1.

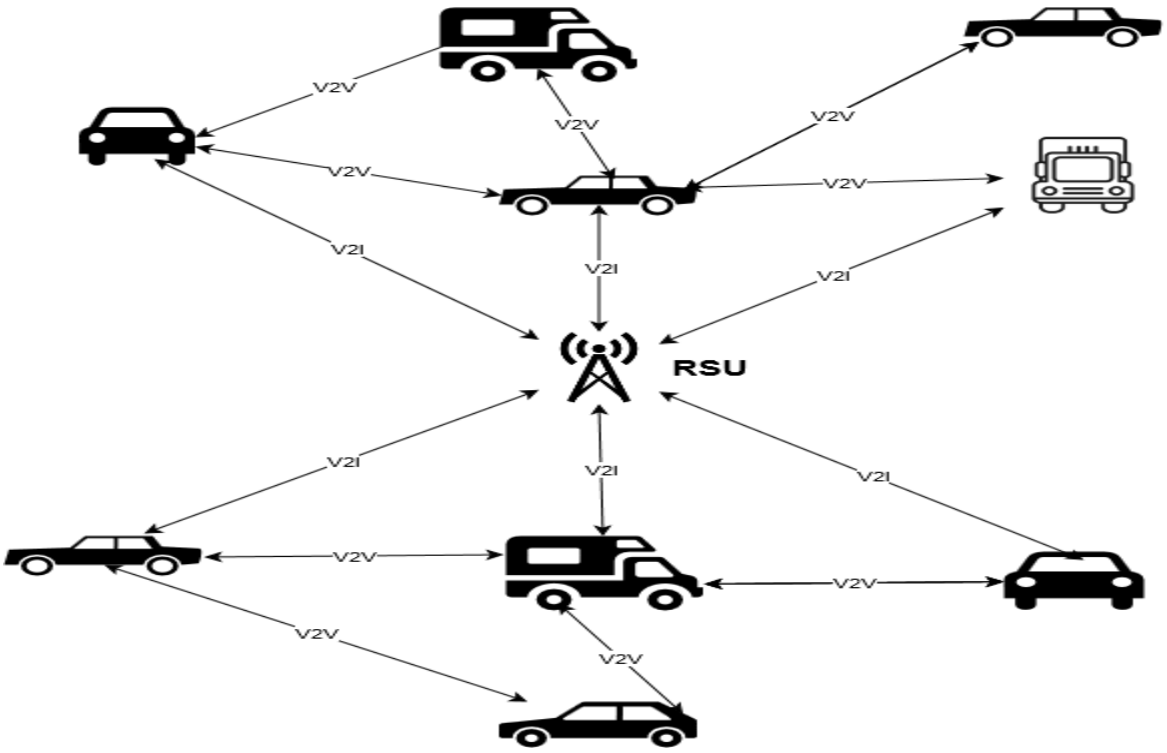


Рис.1.2. Типова архітектура VANET мереж

Згідно макету, у типовій VANET мережі використовуються два типи зв'язків: V2V (Vehicle to Vehicle, ТЗ до ТЗ), V2I (Vehicle to Infrastructure, ТЗ до інфраструктури).

Комунікація між вузлами мережі для усіх типів зв'язків відбувається з використанням бездротового середовища WAVE (Wireless Access in Vehicular Environments), що стандартизоване згідно IEEE.

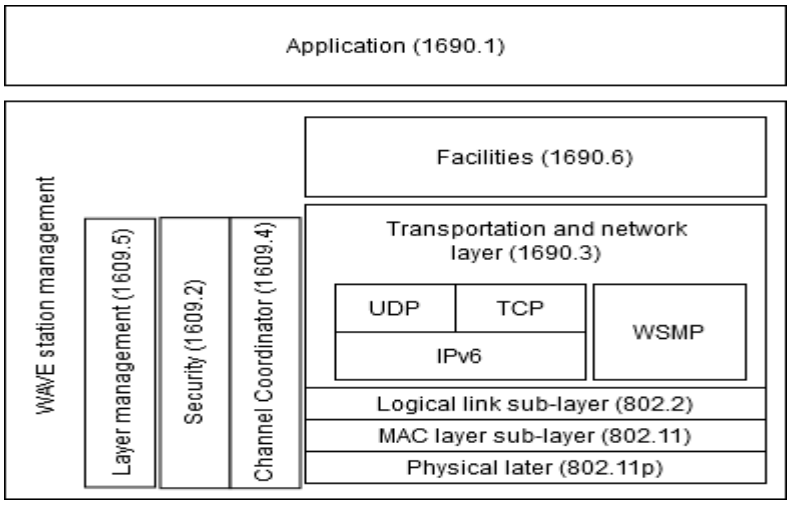


Рис.1.3. Макет WAVE

Типова внутрішня архітектура VANET мережі складається з трьох типів елементів:

- AU (Application Unit) — блок додатку, який розміщений безпосередньо у ТЗ, що виконує функцію забезпечення водія необхідними сервісами та програмними додатками.
- OBU (On-Board Unit) — внутрішній логічний блок зв’язку транспортного засобу, що виконує усі необхідні обчислення та забезпечує зв’язок з іншими вузлами. Типовий OBU має включати в себе антену для радіокомунікації з іншими вузлами; необхідне програмне забезпечення для підтримки стеку спеціалізованих фізичних, канальних та мережевих протоколів; зв’язок з AU для передачі інформації кінцевому користувачу. OBU та AU можуть бути представлені у вигляді єдиного пристрою; в цьому разі поділ є виключно логічним.
- (Road-Side Unit) — придорожній мережевий вузол, що підтримує бездротовий зв’язок з іншими транспортними засобами мережі; зв’язок з іншими RSU може бути як дротовий, так і бездротовий. Даний тип вузлів складається з потужної антени для розповсюдження сигналу, блоку обчислень з відповідним стеком протоколів фізичного, канального та мережевого рівнів, та необхідної інфраструктури для підтримки зв’язку з іншими RSU.

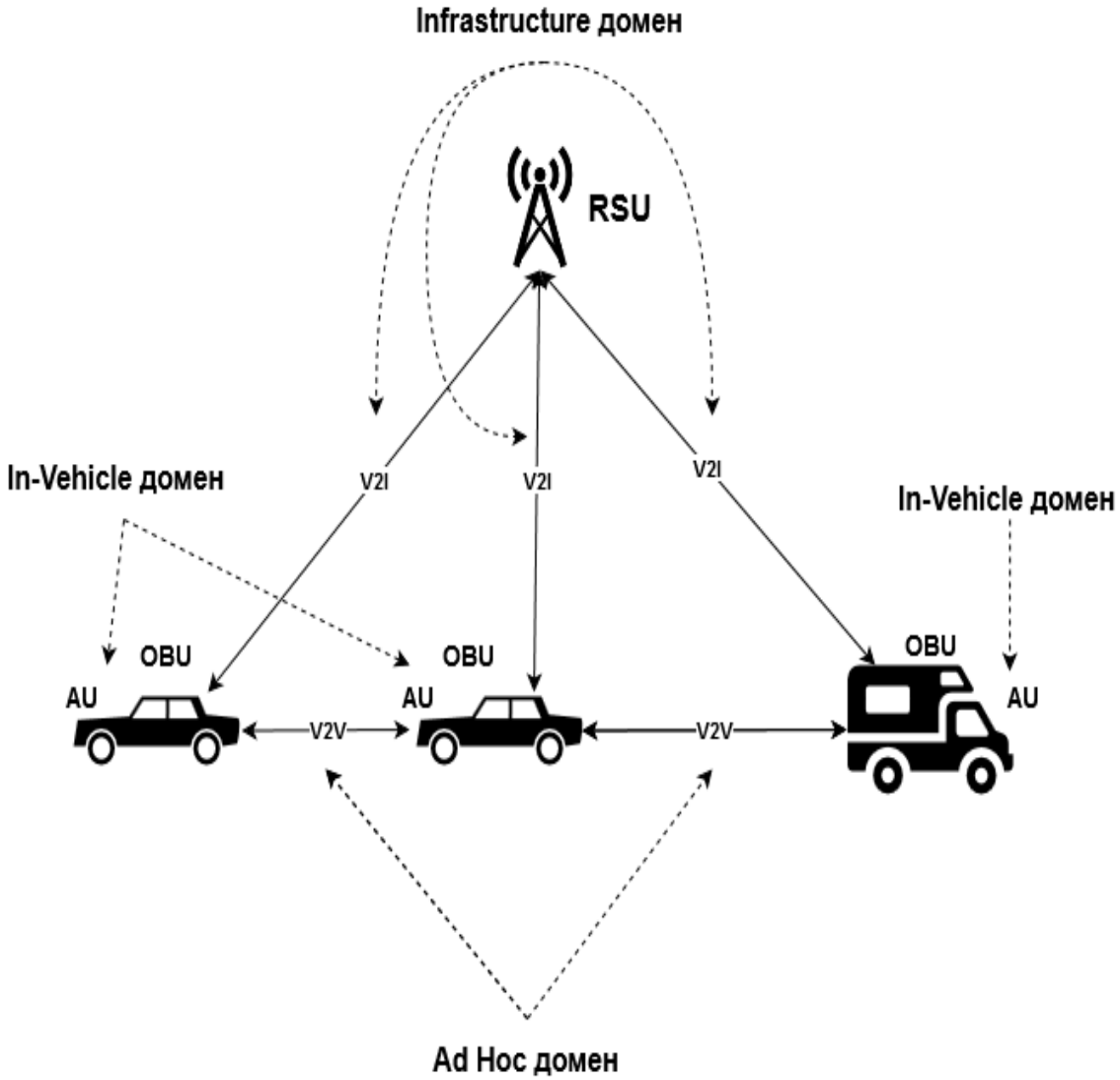


Рис.1.4. Внутрішня архітектура VANET мережі

Транспорті засоби можуть підтримувати зв’язок між собою автономно, без участі RSU, через інтегровані OBU, проте даний підхід значно звужує можливості мережі та використовується виключно при відсутності альтернатив.

Вузли RSU є єдиним статичним елементом класичних VANET мереж. Вибір їх правильного розташування є одним з важливих питань при моделюванні мереж, адже їх розміщення, налаштування та обслуговування суттєво впливає на собівартість мережі. Ці вузли зазвичай встановлюються у місцях великого скупчення транспортних засобів, таких як автостоянки, місця транзиту, перехрестя, головні траси, тощо. RSU виконують функції покращення рівня зв’язку у мережі як статичні вузли з потужним сигналом,

що приймають безпосередню участь у маршрутизації та передачі трафіку, надання послуг зв’язку до Інтернету та відповідних програмних додатків, що можуть бути використані за допомогою цього, тощо.

Таким чином, класичні VANET мережі є децентралізованими та саморегулюючими, адже вони не включають у собі відокремлені сервери та інфраструктуру для обробки інформації, оптимізації маршрутизації, контролю, тощо.

Топологія даного типу мереж є динамічною, що пояснюється постійним створенням та втратою зв’язків між основними вузлами мережі – транспортними засобами. Причиною для цього є постійна зміна їх фізичного місцеположення відносно один одного та інших RSU, що є очевидною характеристикою дорожнього руху, через що зв’язки є недовговічними та постійно змінюються.

Дані особливості внутрішньої структури VANET мереж призводять до значних труднощів у їх практичній реалізації. Особливо складними питаннями є вирішення проблем маршрутизації та безпеки даних користувачів та мережі в цілому через її специфічну топологію, загальнодоступність та відсутність спеціальних централізованих елементів мережі для аналізу та впровадження необхідних політик в залежності від потреб мережі. Практичне вирішення даних проблем для ефективного використання ресурсів мережі та, як результат забезпечення низьких енерговитрат, швидкої передачі трафіку, раціонального виділення радіочастот, тощо, є найбільш важливим для досягнення економічної доцільності імплементації даного виду мереж.

1.3. Протоколи маршрутизації для VANET мереж

На даний момент, VANET мережі налічують велику кількість вже розроблених протоколів маршрутизації для передачі інформації у мережі. Деякі з них є прямо запозиченими з MANET мереж, деякі є їх

удосконаленими варіантами. Також присутні спеціалізовані рішення, що адаптують використання стандартної архітектури VANET для інтеграції певних мережевих технологій.

Вибір правильного протоколу маршрутизації для практичної реалізації VANET мережі є надзвичайно важливим завданням, адже децентралізованість та динамічність структури мережі значно ускладнюють правильний роутинг повідомлень у мережі. Реалізовані протоколи маршрутизації для цього типу мереж поділяються на п’ять груп, що базуються на:

- топології мережі — Topology based protocols;
- геолокації нод мережі — Geo based protocols;
- геоподілу мережі на зони актуальності — Geo-cast protocols;
- кластерному поділу мережі — Cluster based protocols;
- широкомовній трансляції — Broadcast protocols.

Загальний розподіл основних наявних протоколів по групам відповідно до їх характеристик наведено на рис. 1.3 [3].

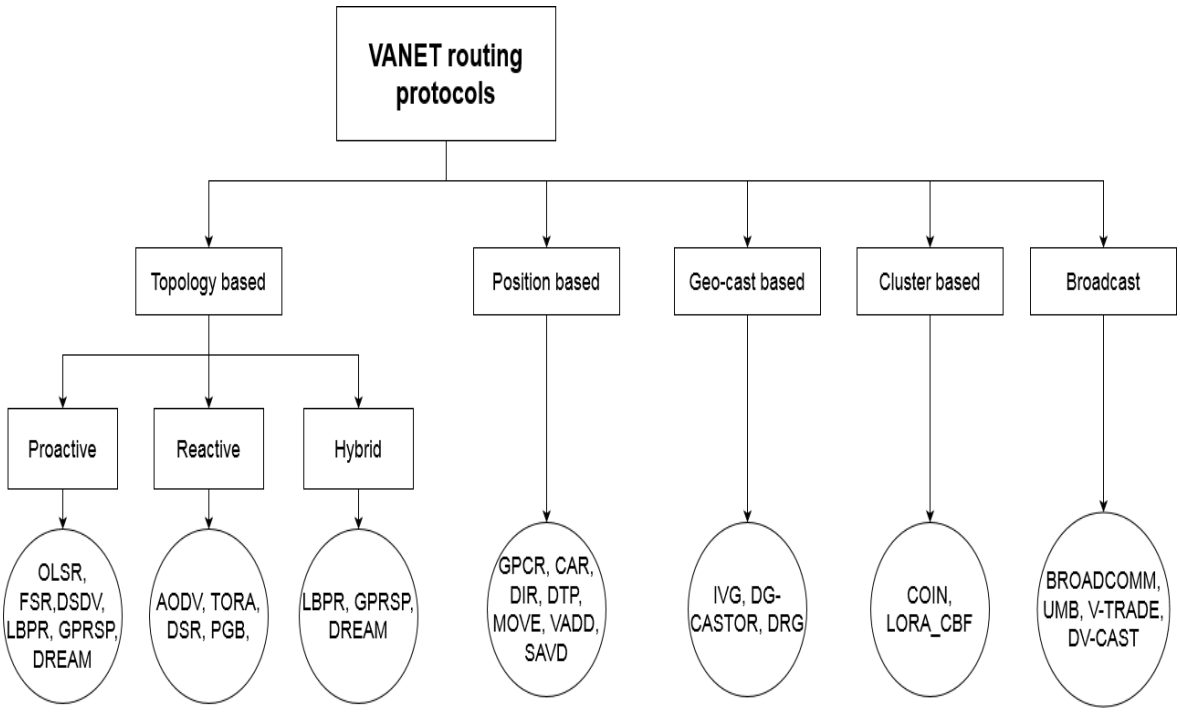


Рис.1.5. Протоколи маршрутизації VANET мереж

Протоколи маршрутизації, що базуються на топології мережі, використовують інформацію про зв’язки між вузлами мережі для пересилки пакетів. Даний тип протоколів поділяється на три групи: проактивні, реактивні та гібридні.

Проактивні протоколи не враховують запити на передачу інформації для роутингу. В даних протоколах таблиця маршрутизації постійно змінюються або оновлюється незалежно від запитів на передачу даних; загальним базисом для створення даних протоколів є BFA (Bellman Ford Algorithm) алгоритм, у якому усі ноди зберігають інформацію, пов’язану з наступною нодою. Таким чином, таблиця маршрутизації кожного вузла зберігає визначені маршрути до усіх вузлів мережі, згідно яких і виконується передача даних при запиті. Розглянемо функціонування даного типу протоколів на прикладі стандартного протоколу — OLSR (Optimized Link State Routing). [4]

Згідно даного протоколу, кожна зміна топології мережі буде переслана до усіх інших нод у мережі, що суттєво підвищує навантаження системи, проте забезпечує кожний вузол миттєвим доступом до найкоротшого шляху до іншого вузла. Контроль мережі відбувається за допомогою двох типів повідомлень: “HELLO” повідомлення та контроль повідомлення. HELLO повідомлення використовуються заради збору даних про зв’язки у мережі та містить таку інформацію: адресу вузла, відправившого повідомлення, його сусідів, їх адреса з вказаним з’єднанням. Цей тип повідомлення сповіщає сусідів відправившого його вузла про усі доступні йому зв’язки, на основі яких кожна нода мережі обирає набір MPR (Multipoint Relay) для подальшої побудови шляху.

Ці повідомлення відправляються через певний проміжок часу; вузли вважаються втраченими, якщо вони не відправляли це повідомлення за вказаний інтервал. Кожен вузол мережі зберігає інформацію про одно- та двох-крокових сусідів.

Другий тип повідомлень — Control message, контрольне повідомлення, — посилається MPR тим нодам, які були обрані в процесі відправлення HELLO повідомлень, та передає інформацію про їх сусідів. На основі отриманої інформації вузли будують граф, що описує мережу, та реалізують таблицю найкоротших шляхів передачі інформації для кожного вузла.

Реактивні протоколи не змушують ноди зберігати інформацію про зв'язки з іншими вузлами системи. Кожен вузол мережі зберігає інформацію лише про однокрокових сусідів та певні; таблиця маршрутизації оновлюється після успішної передачі даних за певним маршрутом; якщо певний шлях не використовується за вказаний інтервал, він очищається, і новий шлях буде збережено після наступної передачі. У разі, якщо повідомлення мусить досягти вузол, шляху до якого немає у таблиці маршрутизації, вузол-джерело перешле пакет усім своїм сусідам, що в свою чергу передають їх далі по зв'язкам за принципом повені доки він не досягне пункту призначення. Отриманий шлях запам'ятовується і зберігається для подальшого використання. Таким чином, мережа не є заповненою постійними службовими повідомленнями, як у проактивних протоколах, та відсутня необхідність зберігання невикористованих маршрутів. Прикладами таких протоколів є AODV, TORA [5].

Гібридні протоколи є спробою поєднання переваг проактивних та реактивних протоколів заради зменшення, відповідно, накладних витрат для збереження маршрутів та затримки початкового пошуку шляху. Основною ідеєю даних протоколів є створення ієрархічних структур-підмереж, взаємодія між якими відбувається за допомогою реактивних методів. Передача повідомлень всередині цих зон відбувається згідно проактивних стандартів. Такі протоколи зазвичай використовуються у разі, коли масштабування мережі є пріоритетною задачею, або коли необхідний компроміс між накладними витратами на комунікацію та затримкою на пошук шляху. Прикладами таких протоколів є ZRP та HARP [6].

Протоколи геолокації з поділами на зони також використовують безпосереднє місцеположення вузла заради побудови так званих зон актуальності та зон посилань. Метою цих протоколів є пересилання повідомлень усім вузлам зони актуальності, тобто використовується алгоритм повені. Вузли, що не відносяться до цієї зони, ігноруються. Прикладами даних протоколів є IVG, DG-CASTOR, DRG [7].

Кластерні протоколи використовуються заради оптимізації великих мереж, що в свою чергу поділяються на підмережі у виді кластерів. Поділ відбувається наступним чином — певна нода обирається як головна нода кластеру, що і відповідає за передачу повідомлень іншим нодам кластеру. Даний підхід дозволяє досягти високого рівня масштабування мережі, проте його використання для моделювання високомобільних мереж є досить затратним з точки зору затримок мережі та накладних витрат. Приклади кластерних протоколів: COIN, LORA_CBF.

Широкомовні протоколи використовуються для пересилання однотипних повідомлень усім вузлам мережі водночас за допомогою алгоритму повені. Зазвичай дані протоколи використовуються для транслювання реклами, сповіщень про надзвичайні ситуації, погоду, дорожні умови тощо. Прикладами таких протоколів є BROADCOMM, UMB, V-TRADE, DV-CAST [8]

Кожна з груп запропонованих протоколів має свої переваги та недоліки; ультимативного рішення-протоколу для побудови довільної VANET мережі на даний момент не існує. Таким чином, остаточний вибір протоколу маршрутизації для побудови конкретної мережі має враховувати наступні фактори [9]:

- розмір мережі та середньостатистична кількість вузлів
- швидкість зміни зв’язків системи, тобто швидкість переміщення транспортних засобів та сила сигналу;
- рівень навантаження мережі службовими повідомленнями

- необхідність збереження даних;
- швидкість оновлення таблиць маршрутизації;
- швидкість реакції мережі на зміну стану;
- використання GPS технологій;
- затримка передачі повідомлень;
- контроль трафіку

Можна виділити 2 основні загальні проблеми даних підходів до маршрутизації у VANET мережах: відсутність можливості динамічної зміни використованого протоколу згідно конкретних потреб мережі, що очевидно змінюються згідно її загальної динамічності; неефективність використання V2I для передачі інформації певних програмних додатків, особливо таких, як відео- та аудіо сервіси, між вузлами транспортних засобів мережі; відсутність рівнів захисту від втручання зловмисників, що можуть легко саботувати роботу мережі та отримати доступ до конфіденційних даних її абонентів.

Таким чином, потенційне рішення даних проблем полягає у пошуку дешевих рішень для імплементації гнучких та потужних централізованих вузлів управління мережею, що дозволили б конфігуративний розподіл ресурсів мережі та забезпечення встановлених політик її безпеки.

1.4 Безпека VANET мереж

Одним з найбільш гострих питань при створенні VANET мереж є захист від зовнішніх та внутрішніх атак. Особливості мереж — динамічність топології, загальнодоступність мережі, повна децентралізація, неможливість використання традиційних методів для забезпечення безпеки — спонукають до знаходження спеціалізованих рішень, які не використовуються у інших типах мереж [10].

Усі типи атак можна поділити на чотири групи

Мережеві атаки — зазвичай мають найбільший пріоритет, адже компромізація безпеки мережі означає, що кожен вузол є потенціальною

жертвою дій зловмисників. Розглянемо стандартні типи атак, що можуть бути вразливими для VANET мереж:

- DOS (Denial of Service, Атака на відмову в обслуговуванні) атака — мережева атака, при якій зловмисник захоплює контроль ресурсами певного транспортного засобу або заклинює канал комунікації VANET мережі. В залежності від масштабності даної атаки та використовуємих протоколів маршрутизації, даний вид атак здібний порушити дієздатність цілої мережі або ж певної її частини. У разі, коли послуги мережі є критично важливими, наприклад, для блокування каналу розповсюдження аварійної інформації чи при автопілотуванні транспорту, такі атаки нанесуть максимальну шкоду.
- Sybil attack (Атака Сивілли) — мережева атака, при якій зловмисник створює велику кількість псевдонимів відносно розміру мережі та передає неправдиву інформацію заради певної мети. Суть атаки полягає в тому, що за допомогою створення імітації великої кількості транспортних засобів, що на правах вузлів мережі мають вагу у прийнятті рішень та передачі інформації, зловмисник маніпулює процесами передачі інформації або ж безпосередньо повідомленнями. Таким чином, наприклад, він може регулювати напрям трафіку через облудні дані про пробки на дорозі за допомогою симуляції даної пробки через псевдоніми.

Атаки на програмні додатки — у даному типі атак зловмисники намагаються маніпулювати даними і контентом певних програмних додатків, які безпосередньо використовуються абонентами мережі. Реальні повідомлення, що надсилаються, змінюються або пригнічуються; таким чином, користувачі або не отримують необхідних даних, або ж приймають сфальсифіковану інформацію. Метою даних атак може бути як і просте

хуліганство, так і забезпечення корисливих інтересів зловмисника. Розглянемо деякі приклади даних атак.

- Атаки фабрикації — у даній атаці зловмисник передає сфальсифіковану інформацію до мережі; фальсифікація відбувається або завдяки безпосередній передачі неправдивих даних, або ж за допомогою імітування зловмисником себе як іншого передатчика, тобто іншого вузла мережі. Цей тип атак включає у собі сфабриковані повідомлення, попередження, сертифікати та персоніфікацію.
- Соціальні атаки — цим типом об’єднані усі атаки, що несуть у собі відкрито провокаційні та образливі повідомлення. Основною метою даних атак є безпосередній вплив на водіїв-учасників мережі та маніпуляція їх поведінкою за рахунок зміни емоційного стану, що може призвести до спалахів конфліктів, хуліганства і т.д.

Моніторинг — тип атак, у яких зловмисники відстежують або збирають певну конфіденційну інформацію про учасників мережі. Шахраї можуть оперувати як група “абонент-аутсайдер”, у якій один із зловмисників є учасником мережі, що передає дані стороннім особам, так і працювати як єдине ціле. Збір інформації відбувається за механікою прослуховування — захоплення пакетів, трансльованих іншими, та їх аналіз на потенційно наявні паролі, сертифікати, особисті дані, тощо. Таким чином, зловмисники можуть відстежувати фізичне місцеположення ТЗ, отримувати доступ до персональних даних водія, тощо.

Слід зазначити, що названі типи атак можуть бути застосовані до обох типів зв’язків у мережі – як до V2V, так і до V2I. Проте забезпечення безпеки передачі пакетів між транспортними засобами та архітектурою є набагато простіше за умови наявності централізованих центрів управління RSU-мережею; покращення рівня безпеки V2V зв’язків потребує розробки та

використання спеціальних протоколів маршрутизації з інтегрованими засобами забезпечення безпеки даних, що негативно впливає на ефективне використання ресурсів мережі; більше того, дані протоколи зазвичай розроблюються з метою запобігання певних видів атак, що залишає мережу вразливою до інших [11].

На даний момент, не існує загального консенсусу згідно стандартів і типових способів забезпечення безпеки у VANET мережах. Доступні рішення включають модифікацію існуючих протоколів маршрутизації, застосування додаткових елементів інфраструктури для впровадження відповідних політик безпеки мережі, використання стандартних засобів для впровадження конфіденційності даних користувачів, таких як симетричні криптографічні алгоритми для шифрування даних, тощо. Слід зазначити, що вищевказані методи зазвичай мають за мету надання захисту від певного типу атак; таким чином, для забезпечення гідного рівня безпеки мережі та персональних даних її користувачів необхідне використання відповідної комбінації запропонованих рішень в залежності від пріоритетів захисту елементів мережі.

1.5. Недоліки та проблеми VANET мереж

Характерні особливості структури VANET мереж природньо викликають складності у їх ефективній практичній реалізації для широкого використання. Основними загальними проблемами є забезпечення безпеки даних користувачів та мережі, ефективне використання ресурсів її вузлів, що зазвичай напряму залежить від обраного алгоритму маршрутизації, стандартизація необхідного радіоспектру задля передачі повідомлень, та проблема менеджменту таких складних та динамічно-змінюваних мереж [12].

1.5.1. Проблема безпеки мережі та даних

Незважаючи на те, що VANET мережі є вразливими до великої кількості атак різного рівня, на даний момент не існує узагальнених рішень згідно забезпечення безпеки учасників мережі та їх даних. Більше того, існуючі методи зазвичай мають на увазі забезпечення безпеки відповідно до певного типу атак, що залишає мережу вразливою до усіх інших. Так як безпеки даних учасників та мережі в цілому є одним з головних пріоритетів при її розробці, особливо у мережах транспортних засобів, де вдала атака може призвести до суттєвих фінансових збитків та нанесення шкоди і потенційної втрати життя людей, є потреба у стандартизації доступних методів захисту мережі та їх комплексного використання для забезпечення гідного рівня безпеки.

Таким чином, проблема впровадження ефективний методів для підтримки безпеки мережі є однією з основних проблем на шляху використання даної технології на побутовому рівні [13].

1.5.2. Проблема ефективної маршрутизації

Незважаючи на велику кількість доступних протоколів та рішень, для ефективного розподілу ресурсів мережі та відповідність її швидкодії та специфікації виникає необхідність ретельного підбору правильного протоколу маршрутизації або ж запровадження нового. Це пояснюється структурними особливостями VANET мереж, що передбачає розподіл нагрузки мережі на вузли без використання відокремлених серверів для аналізу стану мережі та відповідному розподілу навантаження на RSU. Таким чином, виникнення проблем типу bottleneck (вузького місця), коли трафік проходить лише через один вузол, значно сповільнює роботу усієї системи; більше того, враховуючи те, що пропускна здатність вузлів мережі не може бути порівняна з стаціонарними серверами, запобігання проблем такого роду є надзвичайно важливим для ефективного функціонування створеної мережі.

Додатковими проблемами при створенні протоколу маршрутизації для даного типу мереж є запобігання атакам зловмисників, що націлені на роутинг мережі; урахування цього фактору зазвичай сприяє зменшенню ефективності використання ресурсів мережі через інтегрування додаткових кроків алгоритму, націлених на забезпечення безпеки.

1.5.3. Проблема менеджменту мережі

Повна децентралізація класичних VANET мереж не передбачає будь-якого контрольного елементу, що міг би аналізувати стан мережі та приймати рішення про переконфігурування системи на його основі. Таким чином, досягнення високоефективного використання доступних ресурсів мережі є важким завданням, адже розподіл трафіку, визначення протоколів маршрутизації, заходів безпеки і т.д. не можуть бути динамічно змінені у залежності від вимог мережі у конкретний момент. Як наслідок, проектовані мережі або не можуть забезпечувати користувачів гідним рівнем зв’язку під час перевантаження системи, як, наприклад, у денний час-пік міського транспорту, або ж проектуються з надлишковими ресурсами, що значно підвищують собівартість проекту.

Усі вищевказані проблеми є основними перешкодами, що запобігають реалізації технологій VANET мереж на широкомасштабному рівні; таким чином, подальший аналіз доступних рішень та модифікація класичної архітектури мережі на основі сучасних технологій заради їх вирішення і є найоптимальнішим напрямом розвитку даної технології.

Висновки до розділу 1

Проведено аналіз класичних VANET мереж з метою оцінки ефективності потенційних рішень, які дана технологія може запропонувати для покращення стану дорожнього руху та комфорту і безпеки кожного з його учасників.

Розглянуто основні структурні особливості даного типу мереж, такі як типи вузлів та загальні учасники мережі; види зв’язків між ними; топологія мережі. Наведені приклади взаємодії між ними; розглянута внутрішня структура типів вузлів мережі. Проведено порівняння рішень у сфері налагодження маршрутизації даних мереж. Наведені приклади типових атак на VANET мережі та їх основні вразливості; описані основні методи забезпечення безпеки.

Підбито підсумки щодо основних недоліків VANET мереж, що перешкоджають їх використанню на повсякденному рівні та їх потенційних шляхів вирішення.

РОЗДІЛ 2
РОЗРОБКА VANET МЕРЕЖІ З ІНТЕГРАЦІЄЮ SDN

2.1. Аналіз технології SDN

Традиційні рішення для побудови VANET мереж, на жаль, не відповідають сучасним вимогам у сфері безпеки та ефективності використання ресурсів отриманої мережі. Таким чином, для вирішення проблем, сформульованих у першому розділі даної роботи, необхідно використовувати новітні рішення у розробці архітектури мереж.

Так як основними перешкодами до безпосереднього використання VANET мереж пересічним користувачем є недостатній рівень безпеки існуючих рішень, низька ефективність регулювання трафіку внаслідок складності наявних протоколів маршрутизації та відсутність результативних способів моніторингу та адміністрування утвореної мережі. Таким чином, однією з потенційних технологій, що можуть забезпечити суттєві покращення у проектуванні VANET мереж, є SDN – концепт програмно-конфігурованої мережі [14].

Ключовими принципами даного підходу є:

- відокремлення рівнів передачі даних та контролю над ними — таким чином, відбувається відокремлення рівня управління мерею на окремий план, що реалізується програмно;
- створення централізованого управління мережею у програмному вигляді;
- фактична віртуалізація мережі.

Загальна абстрактна структура мережі при використанні даної технології зображена на рис. 2.1.

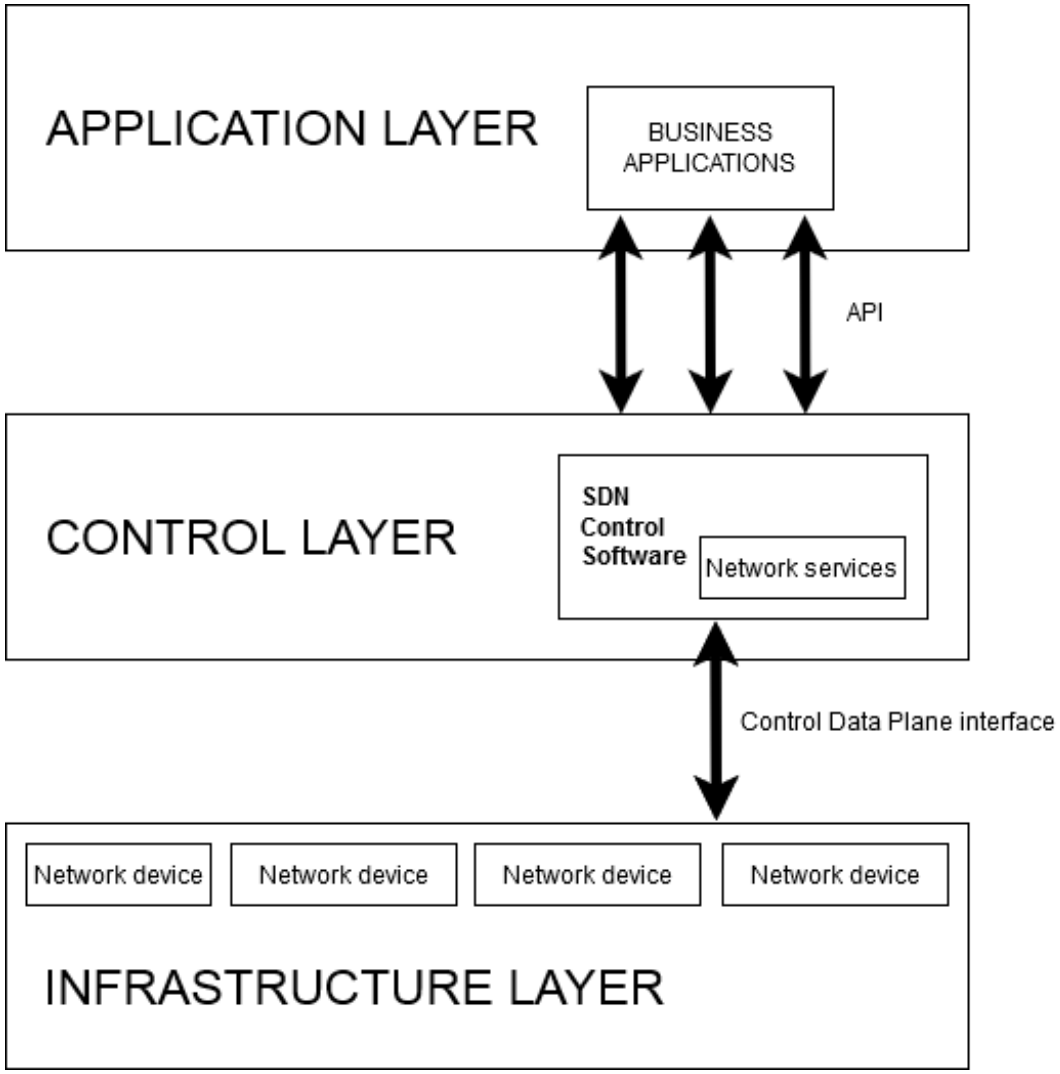


Рис. 2.1. Структура SDN мережі

Так, на відміну від архітектури мереж стандартного типу рівні контролю та інфраструктури (даних) є роз'єднаними. Завдяки цьому значно покращується рівень контролю системи та можливість загального аналізу та зміни її стану на льоту за допомогою програмних засобів замість мануального налаштування кожного елементу мережі.

Управління контролером відбувається за допомогою комунікації між ним та спеціальними SDN програмами, що прямо та точно пересилають свої вимоги до мережі та її поведінки на сам контролер через NBI (northbound interface, північний інтерфейс). Вони також можуть отримувати від контролера дані про мережу у абстрактному форматі для подальшого аналізу та моделювання рішень на основі отриманих даних.

SDN додаток складається з NBI драйверів та безпосередньої логіки програми; дані додатки можуть бути пов’язані між собою за допомогою додаткових NBI, що дозволяє досягти навіть більшого абстрагування контролю мережі.

Сам контролер являє собою центральний елемент SDN мережі і, по факту, є “мізками” системи — його робота полягає у обробці та оформленні задач, що посилаються SDN-додатками, до інфраструктурного рівня мережі для виконання та забезпечення додатків необхідною інформацією про стан мережі. Даний контролер зазвичай складається с одного чи декількох NBI агентів, блоку з описаною логікою контролю та CDPI (Control to Data-Plane Interface) драйверу.

Зв’язок між інфраструктурою мережі та спеціальними SDN контролерами, що і являють собою рівень контролю у даній моделі мережі, відбувається за допомогою певного протоколу; зазвичай даним протоколом є OpenFlow; також використовується протокол OVSDB (Open virtual switch database); деякі потенційно корисні протоколи знаходяться у розробці (i2rs, Interface to the Routing System).

Суть протоколу OpenFlow полягає у створення архітектури типу «контролер-комутатор», у якому роль комутатора полягає виключно у виконанні правил, що задаються безпосередньо контролером мережі. Таким чином, забезпечується гнучкість налаштування правил передачі трафіку та сконцентрованість усіх засобів контролю мережі у одному елементі.

Додатковою особливістю використання протоколу OpenFlow є його адаптивність – для створення мережі можна використовувати комутатори різних виробників з різними внутрішніми інтерфейсами, що дозволяє забезпечувати широкий спектр потреб та використовувати єдиний інтерфейс для роботи з ними за допомогою налагодженого контролера.

Структурно OpenFlow комутатор має включати у собі таблицю потоків, що містять правила, за якими виконується пересилка пакетів у мережі, та

захищений OpenFlow канал, що забезпечує комунікацію між контролером та комутатором та дозволяє контролеру змінювати встановлені правила у таблиці.

Механізм роботи комутатора є доволі простим: отриманий пакет аналізується згідно правил, встановлених у таблиці потоків. Якщо пакет відповідає певному правилу з таблиці, комутатор виконує пересилку згідно ньому. Якщо ж правила немає, то даний пакет або пересилається на контролер через захищений канал для подальшого аналізу та встановлення нових правил у таблиці потоків або модифікації старих, або ж відкидається. Контролер може власноруч передавати пакети даного виду за умови, якщо він передасть команду на пересилку цілих пакетів на той комутатор, що їх надсилає.

Загальний вид OpenFlow комутатора зображено на рис. 2.2.

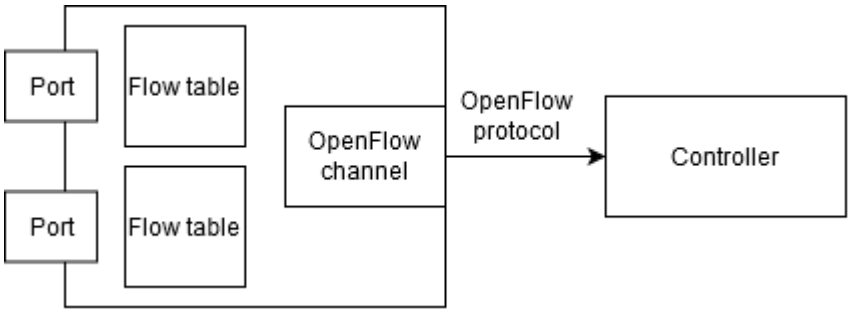


Рис. 2.2. Структура OpenFlow комутатора

Таким чином, центральним елементом SDN мережі є контролер, що відповідає за функцію комунікації програмних додатків для маніпуляції мережі та її фізичним рівнем, що й забезпечує віртуалізацію мережі з усіма її перевагами.

Вищезазначена внутрішня структура SDN мережі продемонстрована на рис. 2.3.

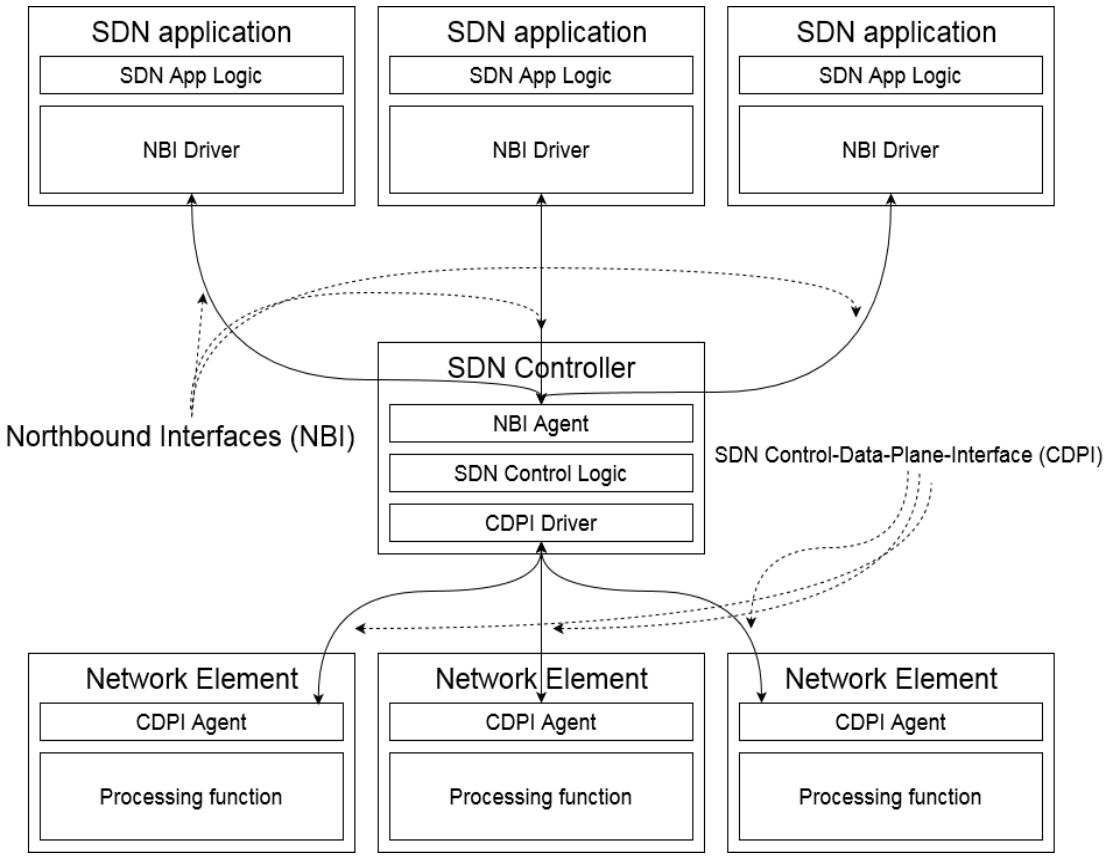


Рис. 2.3. Внутрішня структура SDN мережі

SDN підхід забезпечує низку суттєвих переваг над стандартними архітектурними рішеннями при побудові мереж:

- пряма програмованість мережі та гнучкість її налаштування — за допомогою структури мережі, що використовується, динамічна зміна стану мережі імплементується завдяки комунікації між спеціальними програмними додатками та безпосереднім SDN контролером, що комунікує задані вимоги до мережі на інфраструктурний рівень. Це дозволяє досягати високого рівня ефективності утилізації доступних ресурсів мережі та стрімко вирішувати будь-які проблеми, такі як зміни протоколу маршрутизації або регулювання трафіку при виникненні проблем, наприклад при виникненні ситуації типу bottle-neck;
- централізованість мережі — SDN контролер володіє повним контролем та інформацією про стан мережі, що в свою чергу забезпечує максимальну зручність для аналізу усіх її елементів без

необхідності та розробки відповідного плану дій від програмних додатків.;

- легкість масштабування мережі — завдяки частковій віртуалізації мережі, наявності центрального управління та контролю над неї за допомогою програмних додатків загальне масштабування стає набагато простішим і дешевшим у порівнянні з класичними рішеннями. Проблема надмірного навантаження на контролер може бути вирішена за допомогою створення кластерів контролерів або ж інших ієрархічних систем, що дозволять підтримувати широкомасштабні мережі;
- підвищення рівня захисту системи — централізованість управління та постійний моніторинг стану мережі дозволяє миттєво реагувати на будь-які спроби втручення у її роботу та дозволяє гнучке налаштування політик безпеки на різних рівнях мережі.

Отже, належне інтегрування концепту SDN у класичну архітектуру є потенційно доцільним рішенням щодо покращення вже існуючих варіантів архітектур та як метод боротьби з основними проблемами, що виникають при їх розробці.

2.2. Розробка архітектури VANET мережі з інтегруванням SDN

Так як класичні VANET мережі включають у собі два типи вузлів: транспортні засоби та придорожні блоки зв’язку та має два види зв’язків — V2V, V2I, — є два потенційних шляхи їх розвитку.

Першим шляхом є інтеграція локальних SDN агентів у кожний транспортний засіб та RSU, що будуть об’єднані контролером та SDN комутаторами у єдину мережу. Таким чином, виконається реалізація повного моніторингу стану мережі та централізація елементів її управління у вигляді контролера та програмних додатків, що ним оперують. В такому разі, кожний

транспортний засіб матиме локальний агент, що забезпечуватиме переключення режимів маршрутизації трафіку у мережі між стандартними алгоритмами та передачею даних через OpenFlow комутатори. Це дозволить ефективно розподіляти навантаження на мережу як між доступними RSU, так і транспортними засобами, та забезпечувати доступ до необхідних сервісів, включаючи безпеку мережі та даних її учасників.

Іншим способом розробки архітектури VANET згідно SDN є покращення стану статичної інфраструктури мережі, — RSU вузлів, — за рахунок об'єднання усіх RSU в єдину мережу, якою управлятиме SDN контролер. Виконання цього підходу дозволить позбутися необхідності інтеграції спеціальних локальних агентів у кожний транспортний засіб-користувач мережі, проте забезпечить суттєве покращення контролю трафіку та дозволить централізовано керувати розподілом ресурсів мережі та виконувати моніторинг її стану за допомогою контролеру. Доступ до усіх програмних додатків та їх контенту буде реалізовано безпосередньо через RSU, що забезпечить гідний рівень захисту даних користувачів та мережі.

2.2.1 VANET мережа з локальними SDN агентами

В основу розробки даного типу мережі лежить стандартна архітектура VANET мереж з використанням SDN комутаторів та контролеру для об'єднання усіх пристроїв у єдину централізовану мережу. Таким чином, досягається повний рівень моніторингу стану мережі з її єдиного елементу — SDN контролеру — та встановлення відповідних правил пересилки на кожному з комутаторів в залежності від потреб мережі.

Більше того, створюється зручний та легкий для використання інтерфейс взаємодії між мережею та програмними додатками через північний інтерфейс контролеру, що дозволяє ефективно використовувати ресурси мережі в залежності від вимог програми. Це дозволяє запобігати низці проблем, що виникають при транслюванні великих обсягів трафіку (аудіо, відео

повідомлення) у стандартних VANET мережах та типових ситуацій виду bottle-neck, коли відсутність централізованого управління може спричинити трансляцію трафіку через малопотужний вузол транспортного засобу замість використання доступних ресурсів RSU.

Структура розробленого макету даної мережі зображена на рис. 2.4.

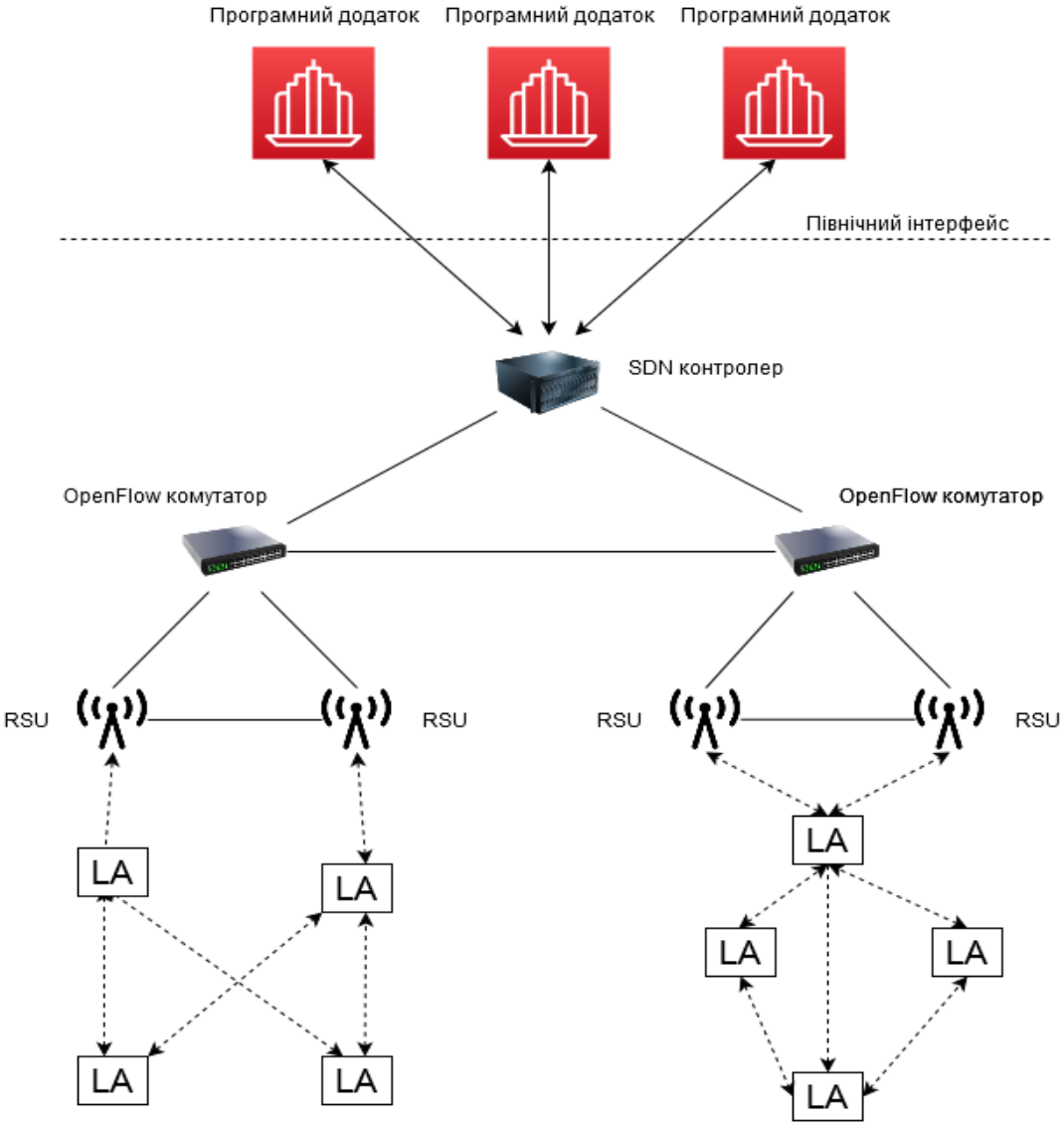


Рис. 2.4. VANET мережа з локальними SDN агентами

На макеті зображено:

- Інтегровані локальні агенти транспортних засобів, що є учасниками мережі, з підтримкою переключення режимів маршрутизації від стандартних алгоритмів, що використовуються у VANET мережах (OLSR, AODV, тощо), до виконання правил

передачі пакетів, встановлених контролером на відповідних комутаторах мережі. Таким чином, транспортні засоби використовують виключно ad hoc з'єднання і стандартні алгоритми маршрутизації між собою у випадку, коли зв'язок з RSU занадто слабкий або зовсім відсутній. Якщо ж ця умова не виконується, контроль трафіку та ресурсів мережі виконується безпосередньо за допомогою контролера, що дозволяє ефективне адміністрування мережі через спеціальні програмні додатки. Це дозволить покращити розподіл навантаження на мережу у випадках, коли мережа використовується для передачі великої кількості трафіку, наприклад, при транслюванні відео, та впроваджувати політику безпеки усієї мережі.

- Придорожні блоки зв'язку — розташовані у стратегічно-визначених місцях прилади, що забезпечують зв'язок SDN комутаторів з транспортними засобами. Такі блоки також виконують функції маршрутизатора та точки доступу. Ці блоки можуть бути з'єднані між собою за допомогою як дротового, так і бездротового зв'язку, забезпечуючи цілісність мережі та подальший розподіл навантаження на неї між собою.
- OpenFlow комутатори, що управляються згідно правил, які встановлює SDN контролер. Містять таблицю потоків з встановленими правилами, що оновлюється контролером в залежності від потреб мережі, та безпечний канал зв'язку з контролером для їх управління та передачі невизначених пакетів для аналізу.
- SDN контролер, як логічний центральний інтелект мережі, що повністю контролює її поведінку та забезпечує постійний моніторинг її стану. За допомогою нього встановлюються необхідні правила передачі пакетів на комутаторах, виконується

аналіз її стану та збирається статистика. Поведінка контролера програмно конфігурується в залежності від потреб мережі за допомогою спеціальних програмних додатків та його власного інтерфейсу. Таким чином, він є центральним елементом розробленої архітектури, так як за допомогою нього забезпечується зв'язок між інфраструктурою мережі та програмними додатками, що регулюють її поведінку.

- Програмні додатки, що чітко та прямо сповідомляють свої вимоги до поведінки мережі на контролер через північний інтерфейс — безпосереднє API контролера. Таким чином, виконується розподіл ресурсів мережі у залежності від вимог цих додатків та їх наявності. За допомогою такого зв'язку може використана інтеграція платформ оркестрування, таких як Puppet, Chef, тощо, або ж спеціальних програмно-визначених засобів захисту та контролю мережі, таких як фаєрволи або додатки, що спеціалізуються безпосередньо на підтримці балансу загрузки мережі.

Враховуючи природні особливості VANET мереж, а саме — ad hoc принцип побудови зв'язків між її вузлами, — для реалізації архітектури даного типу необхідно враховувати імовірність відсутності доступу до RSU певним транспортним засобом через слабкість зв'язку, віддаленість, тощо. Для запобігання відсутності зв'язку з мережею такого транспортного засобу необхідна модернізація його локального логічного агенту, що забезпечує зв'язок з мережею. Метою покращення є інтеграція механізму переходу до стандартних протоколів маршрутизації у VANET мережах (OLSR, AODV, тощо) або ж кешування вже встановлених правил на комутаторах на рахунок втрати зв'язку з основним контролером.

Одним з шляхів потенційного впровадження такої динамічної зміни між централізованим та розподіленим контролем є використання розширених

версій OpenFlow протоколу зі збереженням станів. Такі розширення дозволяють ноді інфраструктурного рівня динамічно обирати різні правила пересилки трафіку в залежності від стану мережі у конкретний момент часу (наприклад, згідно зв'язків та вузлів мережі, що є наявними у той момент). Таким чином, за допомогою превентивного зберігання резервної копії правил пересилок мережі у пам'яті її вузлів, а саме — транспортних засобів, — вони можуть динамічно та автоматизовано обирати між використанням таких правил (і, відповідно, односторонньо змінювати свою поведінку у пересилці) та слідуванням правил, що встановлює SDN контролер. Така автоматизація є надзвичайно важливою, адже її використання дозволить суттєво зменшити використання обчислювальних ресурсів та часу на обробку даних на рівні окремих вузлів мережі.

На даний момент не існує чітко визначених стандартів реалізації даного виду локальних агентів або ж детально вивчених прикладів їх розробки. Загальний вид структури прототипів такого агенту та його взаємодії з контролером зображено на рис. 2.5 [15].

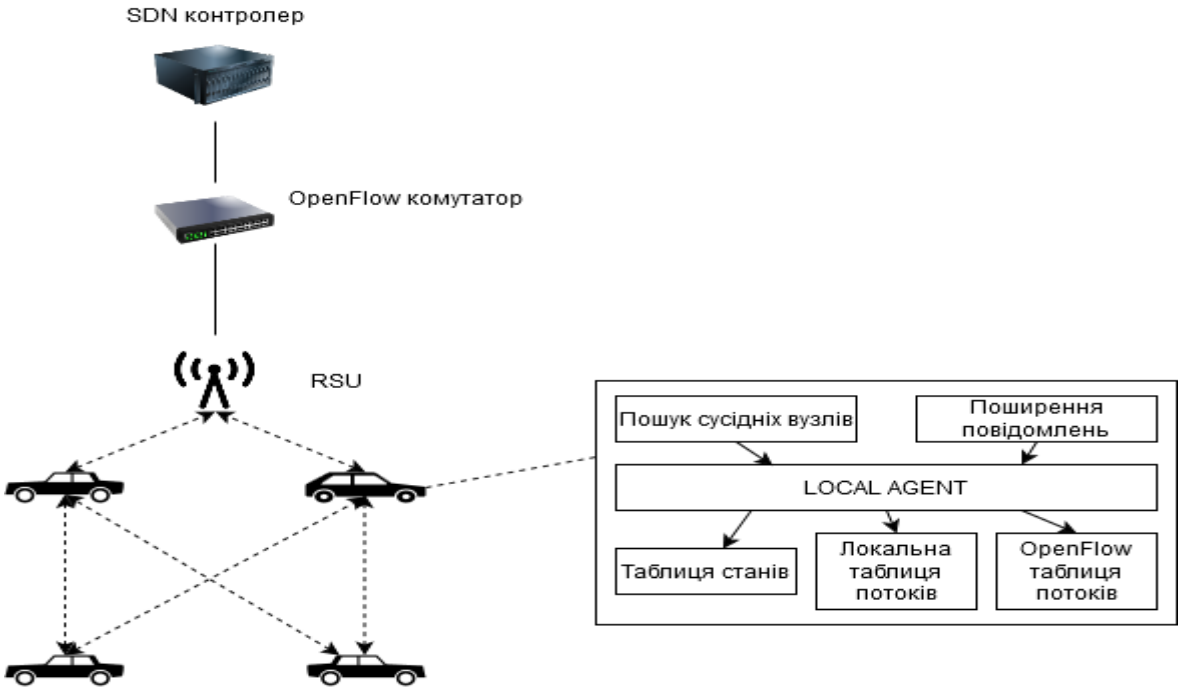


Рис. 2.5. Структура модифікованого внутрішнього блоку зв'язку транспортного засобу

Слід зазначити, що даний підхід не повністю співпадає з реалізацією мережі згідно класичної моделі SDN, адже вузли мережі залишають за собою можливість контролю трафіку у випадку відсутності зв’язку із контролером.

Отже, поділ між рівнем контролю та рівнем інфраструктури не є повним, що не співпадає з загальним концептом програмно-конфігурованих мереж. Проте його використання аргументовано необхідністю наявності запасного механізму маршрутизації для забезпечення потреб мережі у випадку повної недосяжності контролера та можливість уникання додаткових фінансових затрат у випадках, коли розташування інших елементів інфраструктури (RSU, комутатори, тощо) на певних ділянках не є економічно-вигідним. В такому випадку, OBU транспортних засобів просто переключаться на певні наперед визначені протоколи маршрутизації та продовжать своє функціонування.

Таким чином, використання спеціально розроблених локальних агентів у транспортних засобах дозволить модифікувати сучасні варіанти архітектури VANET мереж згідно SDN технології, проте включатимуть у собі резервні копії встановлених правил пересилки даних або ж перехід на стандартні протоколи маршрутизації у разі втрати зв’язку з контролером. Це дозволяє зберегти динамічність утворення зв’язків у мережі та їх самоорганізовуємість, що є надзвичайно важливим у разі їх використання в ситуаціях з відсутністю зв’язку з статичною архітектурою, наприклад, у рятівних операціях.

Основною проблемою запровадження даного підходу є необхідність інтегрування локальних агентів у транспортні засоби для їх функціонування як повноправних членів мережі. На даний момент не існує чітко специфікованих стандартів, згідно яких було б можливе моделювання та створення таких агентів; сучасні рішення знаходяться на рівні прототипів. Більше того, їх використання передбачає модифікацію існуючого протоколу

OpenFlow та забезпечення агентів спеціальними програмними додатками для їх внутрішнього функціоналу.

Додатковою проблемою є відмежування транспортних засобів, що не мають модифікованих OBU від доступу до мережі, що є ще одною завадою на шляху до використання запропонованої архітектури.

Таким чином, моделювання та створення VANET мереж з модифікацією OBU транспортних засобів спеціальними локальними агентами є очевидним напрямком розвитку імплементації технології SDN у мережах транспортних засобів, проте на даний момент цей підхід має велику кількість перешкод для безпосередньої реалізації: відсутність стандартів для розробки модифікованих OBU; необхідність розширення доступних рішень для протоколу OpenFlow або його подібних; експериментальна природа доступних рішень та прототипів; необхідність переоснащення вже вбудованих OBU; необхідність покращення вже існуючих симуляторів мережі або ж розробка спеціалізованих модулів для побудови наведеної архітектури та детального аналізу її ефективності.

2.2.2 VANET мережа з інтегруванням SDN на рівні RSU

Іншим потенційним варіантом імплементації SDN концепту є зосередження зусиль щодо покращення наявних варіантів структури на статичну частину VANET архітектури – придорожні блоки зв'язку (RSU).

Основною перевагою даного підходу є незалежність від інтегрування локальних агентів у кожне з транспортних засобів, що користується мережею; як було розглянуто у макеті попередньо запропонованої архітектури, дана проблема є досить вагомою та потребує детальнішого вивчення.

Навпаки, сучасні рішення дозволяють ефективно впровадити концепт централізованого управління з програмно-конфігурованим контролером на рівні статичної підмережі, що складається з RSU. Це дозволить забезпечити

ефективне та конфігуроване використання ресурсів усіх впроваджених придорожніх блоків зв’язку та дозволить підвищити рівень захисту даних у створеній мережі.

Макет VANET мережі з інтегруванням SDN на рівні статичної архітектури наведено на рис. 2.6.

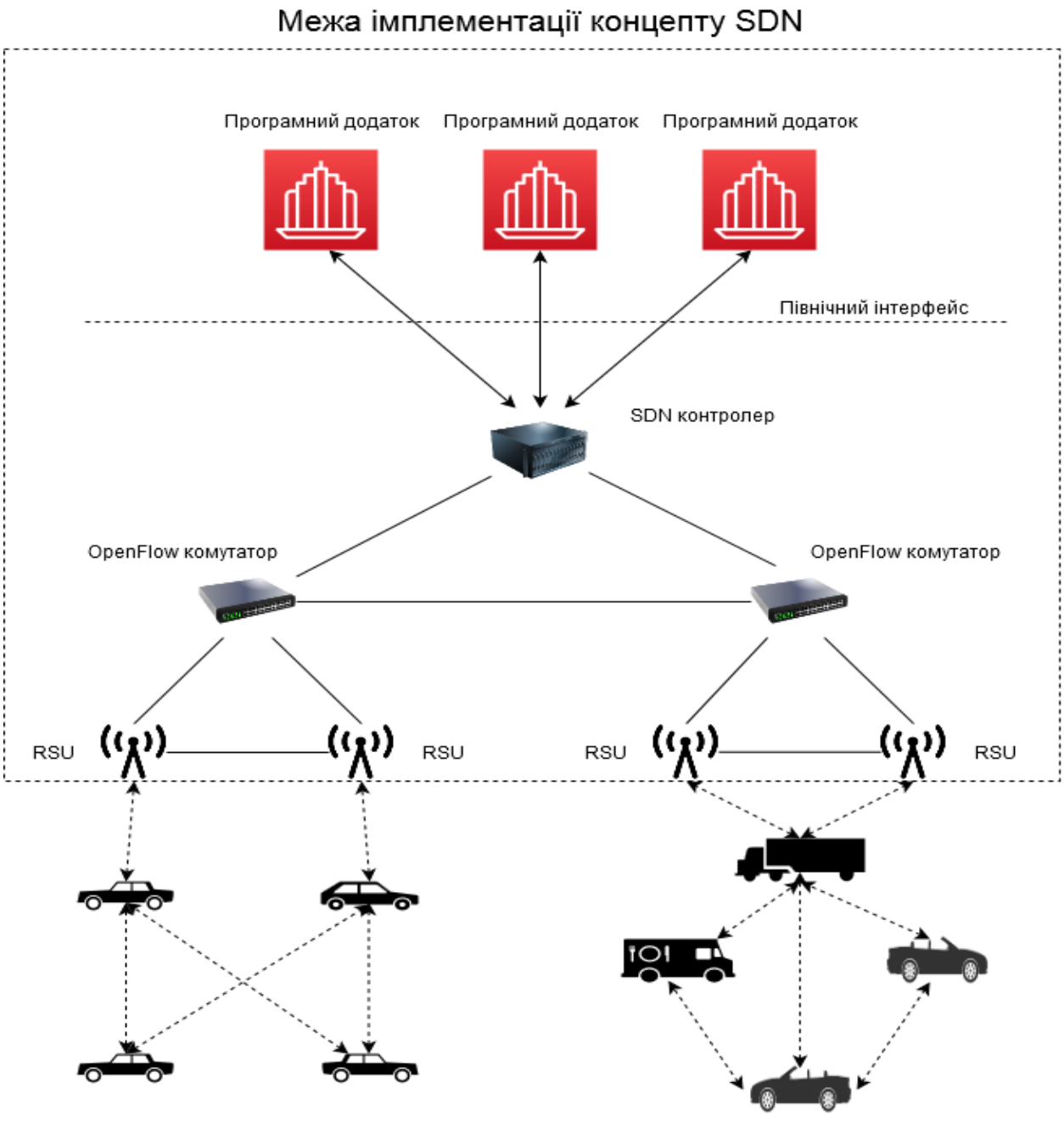


Рис. 2.6. Архітектура VANET мережі з інтеграцією SDN на рівні статичної архітектури

На макеті зображено наступні елементи архітектури:

- Транспортні засоби, що є безпосередніми користувачами послуг, що надаються через RSU. Для взаємодії з мережею кожен з них повинен бути оснащеним вбудованим OBU, структура якого не відрізняється від стандартних реалізацій. Таким чином, спосіб встановлення ad hoc зв'язків між ними у мережі не змінюється. Навпаки, доступ до програмних додатків, якими б могли користуватися водії (наприклад, заради оплачування користуванням платних доріг, розважальні програми, тощо), надається безпосередньо напряму через RSU. За допомогою цього забезпечується захист даних, які використовуються у процесі використання сервісів від втручання зловмисників та протидія їх потенційного використання для деанонімізації, крадіжок, тощо.
- RSU — придорожні блоки зв'язку — що водночас функціонують як маршрутизатори, клієнти мережі та точки доступу. Більше того, саме через них проходитиме трафік усіх програмних додатків, для яких важлива безпека даних користувачів та їх загальна доступність. RSU можуть бути пов'язані між собою та комутаторами за допомогою дротового або бездротового зв'язку. Правильне розташування даних блоків та їх облаштування є суттєвим питанням для інтеграції такої архітектури, адже саме вони забезпечуватимуть доступ користувачів до найбільш важливих сервісів мережі.
- OpenFlow комутатори, що обов'язково містять таблицю потоків з правилами для передачі трафіку, встановленими контролером, та безпечний канал зв'язку з ним. Також можуть бути пов'язані між собою для подальшої структуризації моделі мережі.
- SDN контролер, що виконує розподіл мережевих ресурсів згідно вимогам, які задаються програмними додатками; встановлює правила пересилки на комутаторах, збирає інформацію про мережі

та передає на відповідні програми для подальшого моніторингу та аналізу.

- Програмні додатки, що надають користувачам мережі доступ до певних сервісів, які можуть покращити стан дорожнього руху та рівень комфорту його учасників. Окремо слід виділити спеціальні програми, що повинні забезпечувати безпеку даних учасників мережі та доступу до неї, та програми, що виконують постійний моніторинг стану завантаженості мережі та корегують її стан для його максимального збалансування.

Таким чином, використання SDN концепту у даній архітектурі обмежується взаємодією з RSU і установкою та налаштування контролера та комутаторів. Це полегшує процес інтегрування технології у вже існуючі архітектурні макети та, в цілому, не вимагає суттєвих змін у її структурі.

Незважаючи на те, що певні типи повідомлень, які не несуть критично важливої інформації про учасників або стан дорожнього руху будуть передаватися згідно вже розробленим стандартам та алгоритмам маршрутизації, загальний стан безпеки мережі і ефективність розподілу трафіку значно підвищується за рахунок ізоляції усіх вимогливих сервісів, що використовуються абонентами мережі, та тих додатків, що зберігають або використовують конфіденційну інформацію.

Однією з інших переваг використання SDN підходу до проектування архітектури мережі є легкість її подальшого масштабування до вимог користувачів у порівнянні з традиційними рішеннями. Так, наявність центрального блоку управління мережею — контролеру — дозволяє постійно досліджувати стан мережі і встановлювати висновки щодо ефективності її роботи згідно отриманим даним. У випадку, коли будь-які з елементів не зможуть витримувати навантаження, потенційним рішенням є пряме впровадження додаткових вузлів такого типу — RSU, комутатори. Навіть

контролери можуть бути об’єднані у випадку необхідності та використані у вигляді кластеру.

Забезпечення постійного зв’язку RSU з комутаторами мережі та, відповідно, контролером є досить важливим для підтримки функціонування розробленої архітектури. Тому рекомендується встановлення дротового зв’язку з комутаторами мережі. Потенційно, дана проблема також може бути вирішена за допомогою додаткових модулів з резервними таблицями потоків або алгоритмами маршрутизації для існуючих комутаторів та RSU, проте у такому разі сервіси втрачають рівень захищеності, що забезпечується за допомогою програмних додатків, і стануть вразливими до атак зловмисників.

Головними проблемами даного підходу є визначення стратегічно-ефективних місцеположень для встановлення RSU; розробка спеціального програмного забезпечення для моніторингу, аналізу та розподілу навантаження системи на доступні вузли зв’язку та впровадження певних політик безпеки мережі; відсутність суттєвого покращення передачі та захисту даних у класичному варіанті функціонування — ad hoc зв’язках.

2.2.3. Переваги та недоліки запропонованих архітектур

Загальне порівняння характеристик та принципів використання запропонованих варіантів архітектур VANET мереж з інтеграцією SDN приведено у таблиці 2.1

Таблиця 2.1. Порівняння VANET архітектур

Характеристика	Класична VANET мережа	SDN VANET з локальними агентами	SDN VANET з централізованими RSU
Централізоване управління	—	+	+
Сфера інтегрування SDN	—	Уся мережа за рахунок інтеграції локальних агентів у ТЗ та RSU	Об’єднання та централізація усіх RSU мережі
Маршрутизація	Стандартні протоколи маршрутизації	Контроль над маршрутизацією усієї мережі виконує SDN контролер	Контроль над маршрутизацією трафіку, що проходить через RSU, виконує SDN контролер
Наявність резервного механізму функціонування	—	В кожний ТЗ інтегровано локальний агент для переходу на стандартний Ad hoc режим функціонування у разі втрати зв’язку з контролером	При втраті зв’язку з контролером RSU більше не надають доступ до спеціальних сервісів

Таблиця 2.1. Порівняння VANET архітектур

Характеристика	Класична VANET мережа	SDN VANET з локальними агентами	SDN VANET з централізованими RSU
Масштабування (інфраструктура)	Ускладнено через неможливість централізованої оцінки рівня роботи мережі; потребує налаштування кожного RSU окремо	Необхідність масштабування може бути оцінена через центральний вузол; налаштування окремих RSU є автоматичним через контролер	Необхідність масштабування може бути оцінена через центральний вузол; налаштування окремих RSU є автоматичним через контролер
Масштабування (ТЗ)	Нові абоненти мережі повинні мати інтегрований OBU	Існуючі OBU мають бути модифіковані локальними агентами для доступу до мережі	Нові абоненти мережі повинні мати інтегрований OBU
Безпека мережі та даних	—	Встановлення політики безпеки через програмні додатки для V2I та V2V зв’язків	Встановлення політики безпеки через програмні додатки для V2I зв’язків

На основі порівняння можна зробити висновок, що архітектура SDN VANET з локальними агентами є найперспективнішим шляхом розвитку

даної технології, проте відсутність узгоджених стандартів для реалізації її окремих елементів та необхідність їх удосконалення для забезпечення потреб мережі роблять її не готовою для використання на даний момент. Підхід до інтеграції локальних агентів у MANET мережах, на яких і основуються VANET, знаходиться у стані прототипів для спеціальних потреб державних служб, у т.ч. для тактичних операцій військових служб. Таким чином, її розвиток для громадського використання та, відповідно, перенесення встановлених стандартів реалізації до VANET потребує часу.

Навпаки, архітектура SDN VANET з централізованими RSU може бути реалізована на основі вже існуючих технологій, адже додаткові елементи мережі, які запропоновані в ній, вже реалізовані у вигляді OpenFlow комутаторів та SDN контролерів різних виробників: Cisco, Nicira, Juniper, та багато інших. Таким чином, усі переваги, що надає SDN підхід до проектування мереж, можуть бути використані уже на даному етапі їх розробки.

Використання запропонованої архітектури, базованої на SDN, дозволяє покращити стан основних проблемних характеристик традиційних VANET мереж та удосконалити їх сильні сторони:

- Покращення маршрутизації мережі через наявність центрального вузла контролю, що гнучко налаштовується та має повний огляд стану мережі дозволяє уникати проблем типу bottle-neck або виникнення петель, що є доволі розповсюдженими у класичних VANET архітектурах.
- Забезпечення безпеки мережі та персональних даних її абонентів за допомогою спеціальних програмних додатків для надання інструментів адміністрування мережею та впровадження типових видів захисту, таких як фаєрвол.
- Регуляція використання ресурсів RSU та транспортних засобів — за рахунок моніторингу стану мережі, SDN контролер може

регулювати потужність сигналу кожного RSU та OBU в залежності від навантаження. Таким чином, можна зменшити знос обладнання та енерговитрати у періоди мінімального навантаження на мережу, наприклад, уранці;

- Динамічний вибір використовуваних радіочастот RSU у залежності їх завантаженості.
- Забезпечення централізованого контролю над мережею та можливість їх гнучкого менеджменту та адміністрації за допомогою спеціальних програмних додатків.

Отже, розроблена архітектура надає шляхи вирішення типових наявних проблем VANET мереж та покращує їх загальну ефективність для подальшого громадського використання.

Головними проблемами для її безпосереднього впровадження є забезпечення безпеки центрального вузла системи — контролера, вибір належних комутаторів та контролеру в залежності від їх конфігурованості та розробка програмних додатків для взаємодії з ними.

2.3. Розробка програми для захисту від DDOS атак у SDN VANET

Для наглядної демонстрації потенціалу використання наведеної архітектури необхідне моделювання створеної мережі та розробка певного програмного додатку для взаємодії з нею.

Симуляція роботи SDN VANET із зазначеною архітектурою може бути виконана за допомогою сучасних мережевих симуляторів, що включають у собі модельні реалізації сучасних OpenFlow комутаторів та інтерфейс для інтеграції вже існуючих та використовуваних SDN контролерів.

Так як основною перевагою SDN мереж є їх гнучкість до налаштування, що пояснюється їх програмною конфігурацією, для наглядної демонстрації потенціалу використання наведеної архітектури необхідно розробити

програмний додаток, що буде взаємодіяти із симульованою моделлю даної мережі та виконуватиме певні корисні функції.

Вибір функціоналу розроблюваного додатку базується на забезпеченні основних потреб мережі. Через те, що у традиційних VANET мереж низький рівень безпеки мережі та даних її користувачів, доцільним є розробка програмного додатку для впровадження захисту від певного типу атак на мережу та користувачів.

Згідно теоретичного матеріалу, наведеного у першому розділі, VANET мережі є вразливими до широкого спектру атак. Одним з типових видів атак, що можуть використовуватися зловмисниками, є DDOS атаки для саботажу роботи сервісів мережі.

Основними вузлами, до яких може бути використана DDOS атака у наведеній архітектурі є вузли статичної архітектури – RSU, комутатори, безпосередньо контролер. Це реалізується за допомогою нагромадження великої кількості постійних безглуздих повідомлень, що пересилаються для обробки на вказані вузли мережі, та відповідне її завантаження. Такий підхід гарантує неможливість використання стандартних сервісів мережі її абонентами через банальну нестачу ресурсів, що, з урахуванням особливостей VANET мереж, може призвести як до певного дискомфорту її користувачів, так і до суттєвих матеріальних збитків та потенційної фатальних наслідків по відношенню до людського здоров’я та життя.

Типовими методами захисту мереж від такого виду атак є аналіз трафіку мережі та налаштуванням відповідного брандмауера для його фільтрування.

Вирішення даної проблеми є надзвичайно важливим для забезпечення безпечності використання наведеної архітектури та являється прозорим прикладом легкості та гнучкості налаштування адміністрування та контролю розробленої SDN VANET мережі.

Отже, як доказ ефективності розробленого концепту необхідна розробка програмного додатку для захисту від DDOS атак у запропонованій мережі.

Загальна схема запропонованої системи та її взаємодії з мережею приводиться на рис. 2.7.

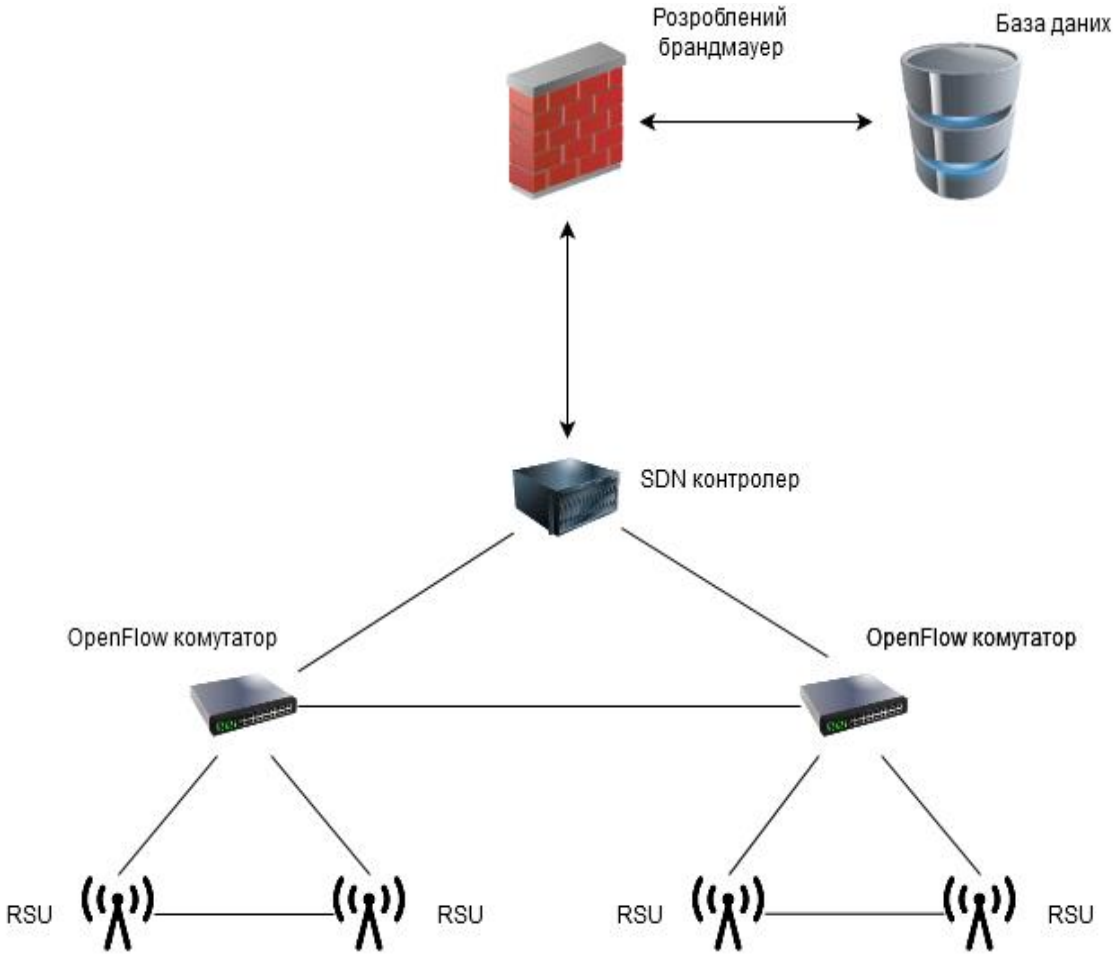


Рис. 2.7. Схема взаємодії системи

Опис взаємодії між елементами системи:

- Інформація про стан трафіку мережі та його характеристики зберігається на SDN контролеру; він передає цю інформацію на брандмауер через північний інтерфейс для подальшого аналізу та впровадження запобіжних дій у разі спроби саботажу системи;
- Брандмауер містить опис логіки пошуку втручень у роботу системи зі сторони зловмисників та включає у собі інтегровані засоби для налагодження зв’язку з контролером. При знаходженні спроб DDOS атак брандмауер автоматично передає команду на заборону користуванням мережею визначеного вузла на певний

період часу та передає запит на збереження інформації про випадок до бази даних.

- База даних отримує інформацію про ідентифікацію вузла, що був учасником DDOS атаки, та зберігає її для подальшого аналізу або впровадження відповідних дій зі сторони адміністрації у випадку рецидиву.

Таким чином, розроблена структура мережі створює ефективні інструменти для її адміністрування та автоматичного захисту від DDOS атак зловмисників. Збереження записів про такі випадки дозволяє виконувати подальший аналіз стану захищеності мережі, визначати ідентифікацію осіб, що намагалися її порушити, та динамічно змінювати політику безпеки в залежності від її вимог.

Загальний алгоритм внутрішньої логіки роботи брандмауера зображено на рис. 2.8.

Рис.2.8. Блок-схема алгоритму роботи брандмауера

Як продемонстровано на блок-схемі, наявність централізованого джерела інформації про стан трафіку у мережі — контролеру, — який також може отримувати дані про стан будь-яких комутаторів, дозволяє захистити мережу від DDOS атак флуд-характеру; отримані дані можуть бути подалі

збережені та аналізовані для впровадження більш ефективних рішень щодо безпеки мережі.

Окрім того, додатковим фунціоналом брандмауера повинно бути забезпечення зручного графічного інтерфейсу користувача для взаємодії між базою даних та контролером у випадку необхідності мануального налаштування контрольного списку доступу (ACL) адміністратором мережі та внесення будь-яких змін у дані для, наприклад, подовження періоду блокування або його зняття.

Таким чином, розроблена система дозволить зручне адміністрування мережі та ACL контролера та забезпечить автоматизований захист від DDOS атак.

Висновки до розділу 2

Розглянуто концепт SDN підходу до розробки мережевої інфраструктури та його ефективність при інтеграції у традиційні VANET мережі.

Запропоновано два варіанти побудови архітектур SDN VANET відповідно до спектру використання технології з повним описом елементів структури та їх взаємодії. Проведено порівняння між отриманими макетами, визначені їх сильні та слабкі сторони, визначені основні шляхи розвитку.

Обраний остаточний варіант архітектури для дослідження та моделювання.

Розглянуто основні типи програмних додатків для взаємодії з мережею в залежності від їх необхідності. Обрано розробку брандмауера із захистом від DDOS атак для демонстрації переваг використання SDN підходу при проектуванні мереж.

Описано загальний макет утвореної системи та внутрішню логіку брандмауера.

РОЗДІЛ 3
МОДЕЛЮВАННЯ І ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1. Вибір симулятору мережі

Для демонстрації результатів роботи запропонованого програмного додатку необхідно забезпечити симуляцію розробленої архітектури VANET з інтегруванням SDN на рівні статичної інфраструктури. Більше того, необхідно переконатись, що запропоноване рішення може бути перевірено та змодельовано за допомогою вже існуючих рішень, тобто практично використованих контролерів та комутаторів, віртуальні моделі яких зазвичай вже запроваджені у відповідних емуляторах.

На даний момент існує достатньо широкий вибір мережевих симуляторів, що дозволяють емулювати роботу SDN мереж та забезпечують кінцевих користувачів необхідними розробленими елементами таких мереж як широко використованих пакетів цих симуляторів.

Прикладами таких симуляторів, що задовольняють основні запити щодо тестування SDN VANET мереж на рівні пакетів даних, є NT-3 та Mininet-WiFi [16]. Порівняння їх характеристик наведено у таблиці 3.1.

Таблиця 3.1. Порівняння мережевих симуляторів

Характеристика	NT-3	Mininet-Wifi
Графічний інтерфейс	+	+
Підтримка сторонніх контролерів	Доступна у вигляді модулів, проте складна у використанні	Безпосередньо інтегрована та вимагає мінімум налаштування
Підтримка OpenFlow	+	+
Важкість розробки	Вимагає налаштування окремих модулів	Є цілісною системою
Важкість користування	Необхідне вивчення інтерфейсу	Простий інтерфейс

Головними характеристикою даних симуляторів є їх направленість на дослідження внутрішньої роботи мережі у абстрагованому середовищі. Проте вони забезпечують імпортування створених моделей архітектури у інші VANET симулятори, які дозволяють користувачу змоделювати поведінку транспортних засобів на реальних дорогах з їх візуалізацією. Таким чином, можливе моделювання поведінки трафіку на реально існуючих дорогах за допомогою їх інтеграції через, наприклад, Google мапи. Прикладом такого симулятору є SUMO.

На основі порівняння характеристик запропонованих симуляторів для емуляції роботи розробленої архітектури вирішено використати симулятор Mininet-Wifi через зручний користувацький інтерфейс та простоту налаштування усіх елементів запропонованої архітектури, таких як комутатори OpenFlow різної версії, SDN контролер з можливістю інтеграції стороннього контролера, безпосередні абоненти мережі – транспортні засоби та придорожні блоки зв’язку.

Більше того, Mininet-Wifi має вбудовану інтеграцію з графічним емулятором транспортних мереж – SUMO – що дозволяє використовувати розроблену архітектуру для перевірки її функціонування в умовах емуляції реального дорожнього руху на певній ділянці дійсних доріг, що моделюються за допомогою Google мап.

3.2. Вибір мови програмування

Для розробки програми брандмауера мережі обрано мову програмування Python. Цей вибір дозволяє легко інтегрувати створений продукт з симулятором мережі Mininet-Wifi, який також написаний на Python та дозволяє емуляцію мережі на рівні пакетів даних за допомогою простих скриптів.

Більше того, використання Python має такі переваги у порівнянні з іншими рішеннями:

- Підтримка ООП
- Портативність
- Розширення с C та C++
- Відносна простота використання
- Низький поріг входження

Таким чином, код програм, написаних на Python, зазвичай є простим для розуміння навіть для тих, хто не має суттєвого досвіду роботи з даною мовою, та є суттєво більш лаконічним у порівнянні з Java або C++. При необхідності максимальної оптимізації швидкодії можливе використання додаткових розширень с C або C++ [17].

Компактність коду також забезпечується завдяки широкому спектру стандартних бібліотек, що значно полегшують подальшу розробку програмних продуктів за рахунок інтеграції стандартних ефективних та оптимізованих рішень.

Також слід зазначити, що Python зазвичай забезпечений зручними API клієнтами від таких важливих компаній, як Google; взаємодія з розробленими SDN контролерами через північний інтерфейс – їх API – також підтримується і вимагає мінімум зусиль для адекватної інтеграції.

На основі усіх вищенаведених переваг мовою для розробки брандмауера для захисту від DDOS атак у SDN VANET мережі обрано Python.

3.3. Вибір SDN контролера

На даний момент доступний широкий вибір SDN контролерів для широкого використання. Кожен з них має набір унікальних характеристик та певну мету для використання. Загальним поділом для доступних рішень визначимо їх доступність: відкритість коду та безкоштовність для використання.

Прикладом доступних контролерів з відкритим кодом є OpenDaylight, ONOS, Project Floodlight, NOX/POX.

Спеціальні рішення, що пропонуються від основних фігур на ринку і не є безкоштовними:

- Nuage Virtualized Services Controller (VSC) від Nokia
- UniFi by Ubiquiti Networks, тощо [18].

Очевидно, що для подальшого моделювання розробленої мережі доцільно використати доступні рішення з відкритим кодом.

Основними вимогами до обраного контролеру є:

- Підтримка масштабування мережі
- Підтримка різних версій протоколу OpenFlow для подальших розробок
- Доступний API для взаємодії з програмними додатками
- Зручний користувацький інтерфейс
- Наявність реалізації внутрішнього ACL та брандмауера для кінцевої взаємодії.

На основі даних запитів обрано контролер Floodlight через простоту його інтеграції за допомогою сервісу для автоматизації розгортання та управління додатками в середовищах з підтримкою контейнеризації Docker та наявному ACL і Firewall із REST API для подальшої взаємодії [18].

3.4. Опис програми

Програмний додаток написаний мовою Python. Для забезпечення графічного інтерфейсу користувача використовується бібліотека Tkinter.

Програма написана згідно стандартів об’єктно-орієнтованого програмування та містить наступні класи: Reader, Writer, Analyzer, Wrapper.

Розглянемо кожний з них окремо:

- Wrapper – основний клас програми, що забезпечує кінцеву взаємодію між розробленими елементами та налаштовує графічний інтерфейс програми.
- Writer – відповідає за зчитування логів контролеру та зберігає їх у текстовому файлі.
- Reader – аналізує таблиці потоків контролеру за допомогою регулярного виразу і передає отримані дані для подальшого аналізу.
- Analyzer – безпосередня реалізація політики безпеки мережі, у якій зазначаються загальний алгоритм мір, що застосовуються до клієнтів мережі, які намагаються порушити її безпеку.

Алгоритм автоматизованої політики безпеки мережі повністю відповідає запропонованому алгоритму з другого розділу роботи, блок-схема якого зображена на рис. 2.8.

3.5. Виконання моделювання роботи мережі та програмного додатку

Моделювання роботи програми та мережі виконується на операційній системі Ubuntu 20.04 LTS.

Для емуляції мережі у симуляторі Mininet-Wifi використовується віртуальна машина з встановленим VM образом, що наданий розробниками симулятора для зручного використання симулятора без необхідності його налаштування кінцевим користувачем.

Запуск контролера Floodlight виконується за допомогою команд у терміналі для взаємодії з Docker. Після того, як контейнер завантажений за допомогою команди «docker pull glefevre/floodlight», необхідно його ініціалізувати за допомогою наступної команди:

Рис. 3.1. Ініціалізація контейнеру контролера Floodlight

У відповідь Docker сповіщує ідентифікаційний номер контейнера. Зазначимо, що у команді визначаються порти для інтерфейсу взаємодії з контролером – 8080, та безпосередній порт власне контролеру – 6653.

Після виконання даної команди можна спостерігати стан контролеру та будь-яким чином взаємодіяти з ним за допомогою вбудованого веб-інтерфейсу. Для цього необхідно перейти за посиланням: http://localhost:8080/ui/pages/index.html, де 8080 – порт інтерфейсу, зазначений у команді вище [20].

Mininet-Wifi дозволяє розробку скриптів з налаштованими топологіями у двох форматах: з використанням графічного інтерфейсу та прямим налаштуванням топології за допомогою безпосереднього написання скрипта.

Наступна топологія, зображена на рис.3.2, була розроблена за допомогою вбудованого графічного додатку для роботи з Mininet-Wifi – Miniedit.

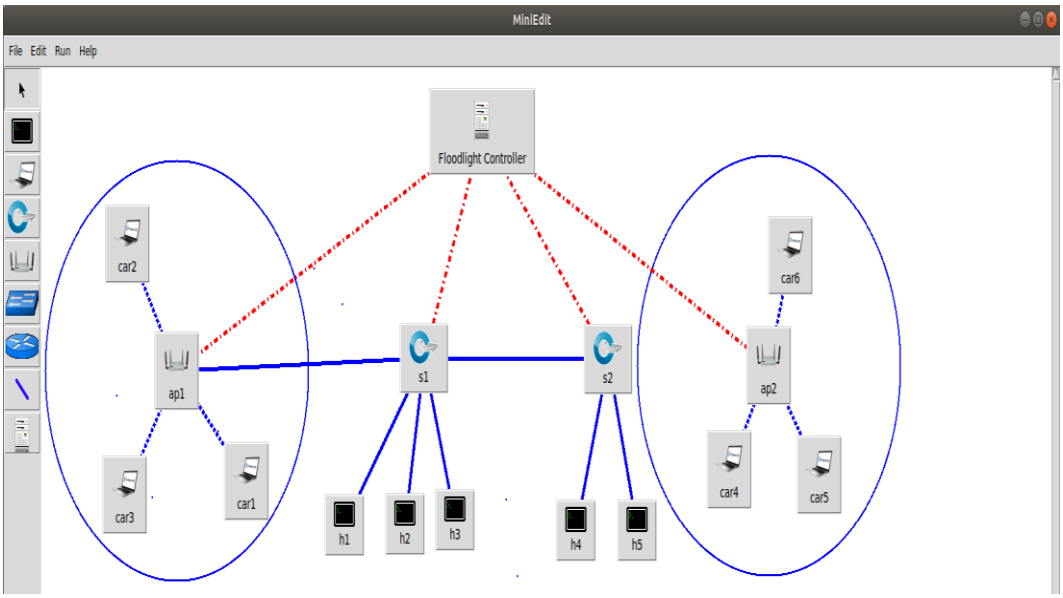


Рис.3.2. Топологія модельованої мережі

На топології зображено:

- Безпосередньо Floodlight контролер, який підключається до вказаної топології через відповідний порт.
- RSU у вигляді AP – бездротових точок доступу. AP мають інтегрований функціонал OpenFlow комутаторів, тому вони можуть напряму під'єднуватись до контролеру та інших комутаторів.
- OpenFlow комутатори – s1-s2.
- Транспортні засоби, які підключені до бездротових точок доступу – car1-car6.
- Додаткові клієнти мережі, які виконують функції серверів програмних додатків, що використовуватимуться у мережі – h1-h5.

Таким чином, дана топологія відповідає архітектурі VANET з SDN на рівні статичної архітектури, запропонованій для аналізу та моделювання у другому розділі даної роботи.

Ініціалізація мережі відбувається згідно стандартного запуску скрипту Python; перевірка вузлів мережі виконується за допомогою команди nodes:

```
wifi@wifi-VirtualBox:~$ sudo python SDN_VANET.py
*** Adding controller
*** Add switches/APs
*** Add hosts/stations
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Add links
*** Starting network
*** Starting controllers
*** Starting switches/APs
*** Post configure nodes
*** Starting CLI:
mininet-wifi> nodes
available nodes are:
ap1 ap2 c0 car1 car2 car3 car4 car5 car6 h1 h2 h3 h4 h5 h6 s1 s2
mininet-wifi>
```

Рис.3.3. Ініціалізація мережі та її вузли

Для демонстрації функціонування елементів мережі виконаємо стандартну перевірку зв’язку за допомогою команди ping:

```
mininet-wifi> car1 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.064 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.025/0.053/0.070/0.019 ms
mininet-wifi> h2 ping car5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=160 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.739 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.047 ms
^C
--- 10.0.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 0.047/53.637/160.125/75.298 ms
mininet-wifi> car1 ping ap1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.082 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.032/0.056/0.082/0.021 ms
```

Рис.3.4. Перевірка зв’язку між вузлами

Відповідна топологія та її елементи відображуються на веб-інтерфейсі контролера.

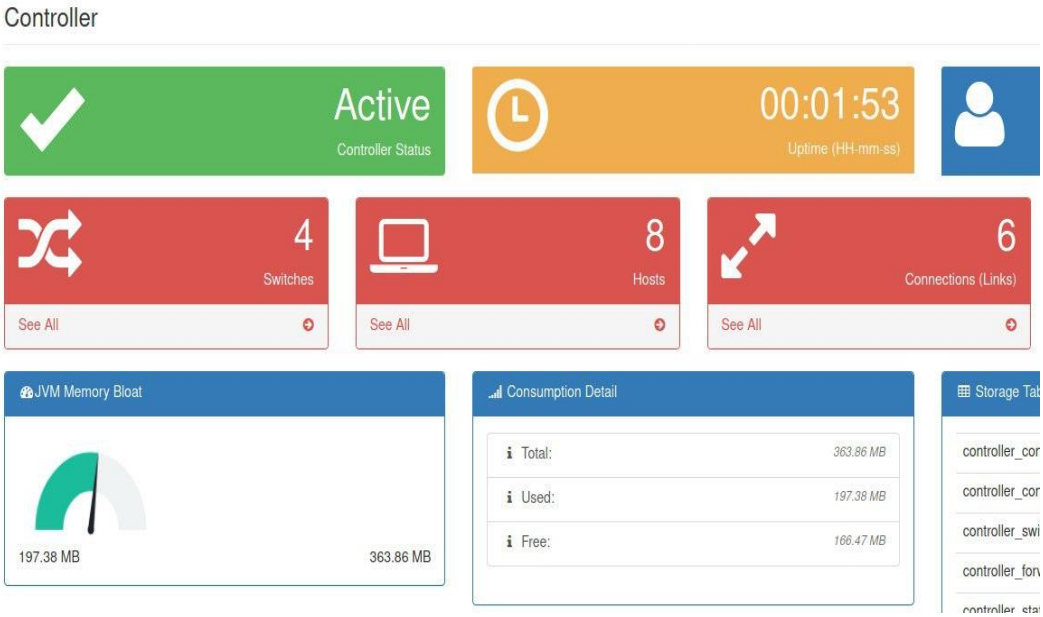


Рис.3.5. Інформація контролера про мережу

Після цього модель запропонованої архітектури можна вважати налаштованою та готовою для тестування роботи розробленого брандмауера.

Запустимо брандмауер, перейдемо до головної панелі та ввімкнемо захист мережі.

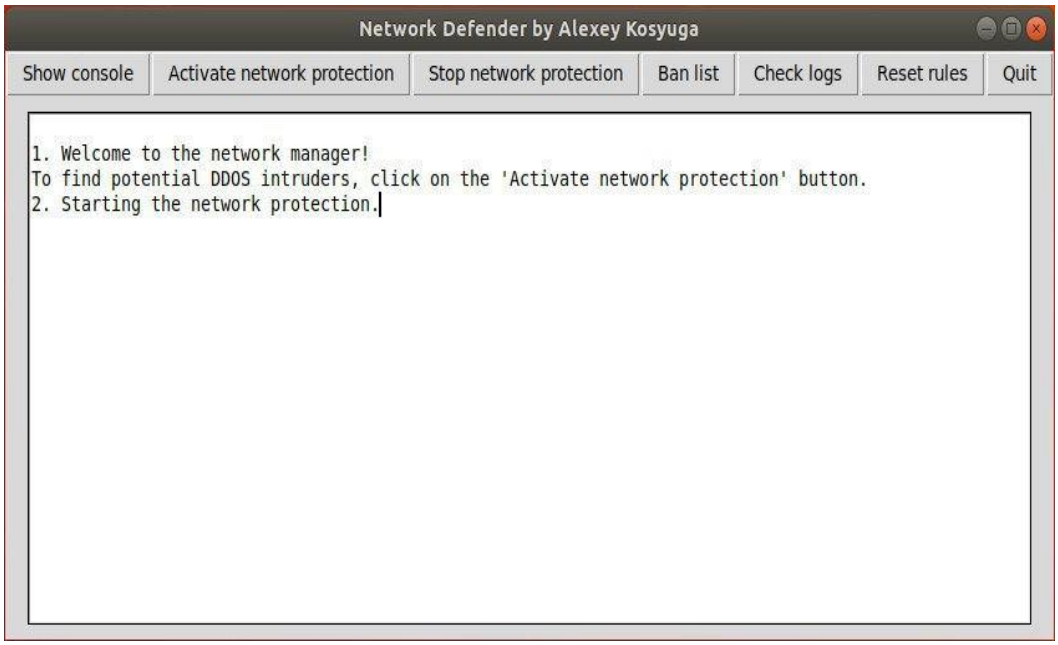


Рис.3.6. Головне меню брандмауера

Виконаємо емуляцію DDOS атаки за допомогою виконання команди “ping -f” у налаштованій симуляції мережі.

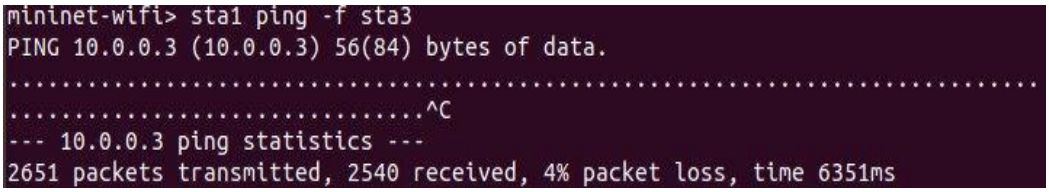


Рис.3.7. DDOS атака у мережі

Так як автоматичний захист мережі вже запущено, брандмауер реєструє спробу DDOS атаки та блокує хоста, що її спричиняє.

Усі дії додатку реєструються у консолі головного меню.

Рис.3.8. Заблокований користувач

Виконаємо перевірку доступу заблокованого користувача до мережі за допомогою команди “ping”.

```
mininet-wifi> sta1 ping -f sta3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
.....
.....
......^C
--- 10.0.0.3 ping statistics ---
164 packets transmitted, 0 received, 100% packet loss, time 2629ms
```

Рис.3.9. Перевірка зв’язку користувача

Як видно з результатів команди, заблокований хост більше не може передавати пакети до інших учасників мережі.

Таким чином, брандмауер успішно обмежив доступ зловмиснику до мережі та забезпечив її безпеку від його подальших втручань.

3.5. Аналіз результатів

Результати тестування роботи мережі та програмного додатку є втішними. Розроблена система дозволяє адміністратору власноруч встановлювати політику безпеки за допомогою передачі відповідних правил на контролер.

Робота автоматичного захисту мережі від DDOS атак працює швидко, затримка між початком атаки та її реєстрацією у брандмауері не перевищує 2-3 секунд. На основі цього можна зробити висновок, що вона є достатньо ефективною для безпосереднього використання у захисті мережі.

Інтерфейс програми дозволяє зручно і ефективно контролювати трафік у мережі та не вимагає знання команд – кожній функції відповідає кнопка або віджет, що забезпечує максимальний комфорт користувача та не вимагає спеціальних знань для використання програми.

Усі дії брандмауера відображаються у головному вікні програми, що дозволяє легко відслідковувати кожний етап його роботи.

На основі цього можна зробити висновок, що розроблений програмний додаток є ефективним рішенням для забезпечення захисту від DDOS атак у розробленій архітектурі SDN VANET.

Таким чином, розроблена система демонструє суттєве покращення у рівні безпеки у порівнянні з традиційними VANET мережами і є ефективною демонстрацією влучності використання SDN підходу для розробки VANET мереж.

Висновки до розділу 3

Розглянуто симулятори мереж для моделювання архітектури SDN VANET. Обрано кінцевий симулятор для розробки топології та її емуляції.

Обрано мову програмування для розробки програмного додатку.

Розглянуто наявні SDN контролери. Визначений контролер для використання для симуляції мережі з інтегрованим ACL та зручним інтерфейсом.

Описано структуру програми та її елементів.

Виконано моделювання роботи розробленої архітектури мережі та брандмауеру.

Проведений аналіз отриманих результатів роботи розробленої системи.

ВИСНОВКИ

Розглянуто симулятори мереж для моделювання архітектури SDN VANET. Обрано кінцевий симулятор для розробки топології та її емуляції.

Робота присвячена удосконаленню VANET мереж за допомогою SDN підходу. Було проведено огляд традиційних рішень VANET архітектур, проведений аналіз їх головних особливостей та характеристик. Виявлено основні недоліки та їх потенційні шляхи рішення.

Проаналізовано SDN підхід до побудови мереж, на цій основі розроблено два варіанти архітектури SDN VANET. Описано основні переваги та недоліки розроблених архітектур, виконано порівняння їх характеристик. Обрано остаточну модель архітектури для симуляції згідно легкості її інтеграції у існуючі рішення.

Визначено основні вимоги до програмного додатку для захисту від DDOS атак у розробленій мережі. Обрано SDN контролер мережі, середовище симуляції та мову програмування для реалізації додатку на основі визначених вимог. Розроблено графічний інтерфейс для зручного користування додатком.

Проведено моделювання і тестування розробленої системи. Проаналізовано ефективність її функціонування та зручності для пересічного користувача.

На основі проведеної роботи визначено, що розроблена система відповідає встановленим вимогам та забезпечує рішення проблеми безпеки VANET мереж.

СПИСОК ЛІТЕРАТУРИ

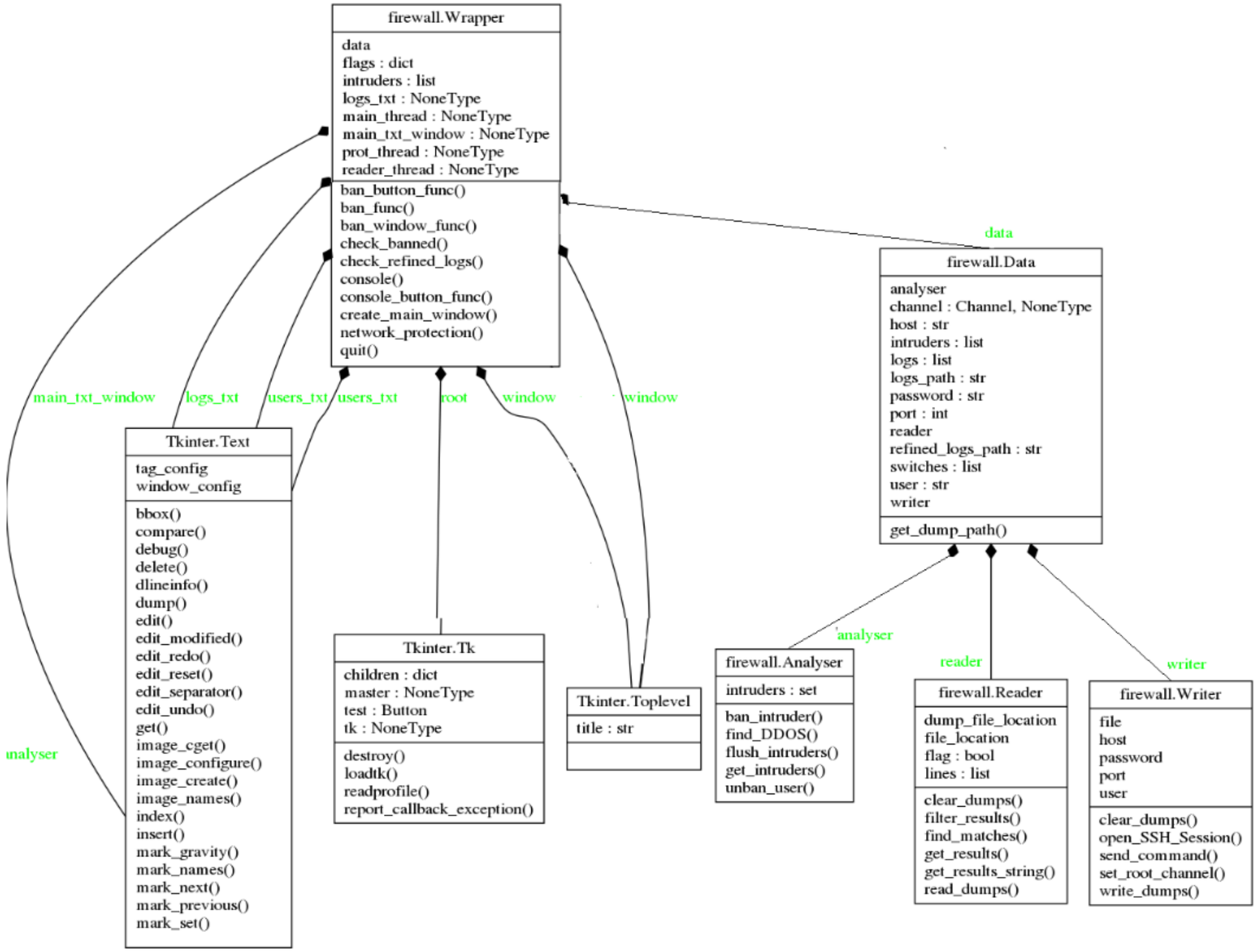
1. Morteza Mohammadi Zanjireh; Hadi Larijani. A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs. [Електронний ресурс] / Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7145650
2. Sommer, Christoph; Dressler, Falko. Vehicular Networking. ISBN 9781107046719.
3. Muhammad Rizwan Ghori. VANET Routing Protocols: Review, Implementation and Analysis [Електронний ресурс] / Режим доступу до ресурсу: https://iopscience.iop.org/article/10.1088/1742-6596/1049/1/012064/pdf
4. T. Clausen, Ed. Network Working Group. Optimized Link State Routing Protocol (OLSR) [Електронний ресурс] / Режим доступу до ресурсу: https://tools.ietf.org/html/rfc3626
5. C. Perkins E. Belding-Royer, S. Das. . Network Working Group. Ad hoc On-Demand Distance Vector (AODV) Routing [Електронний ресурс] / Режим доступу до ресурсу: https://tools.ietf.org/html/rfc3561
6. Zygmunt J. Haas, Marc R. Pearlman, Prince Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks [Електронний ресурс] / Режим доступу до ресурсу: https://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04
7. Neha Goel, Isha Dhyani, Gaurav Sharma. A Study of Position Based VANET Routing Protocols [Електронний ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/312560573_A_study_of_position_based_VANET_routing_protocols
8. N. Wisitpongphan, Fan Bai, O.K. Tonguz. DV-CAST: A distributed vehicular broadcast protocol for vehicular ad hoc networks [Електронний

ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/224132368_DV-CAST_A_distributed_vehicular_broadcast_protocol_for_vehicular_ad_hoc_networks

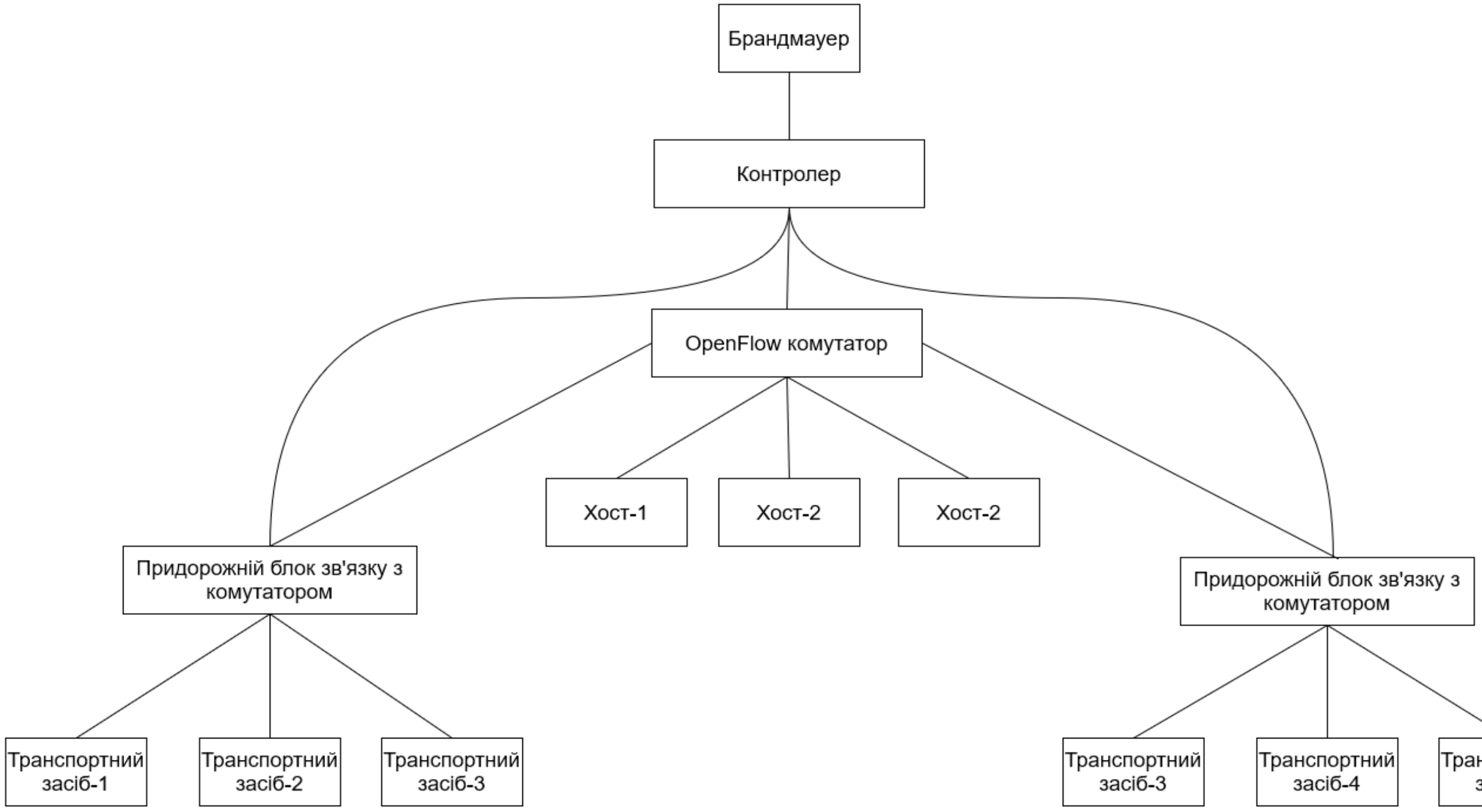
9. Bouchra Marzak, Hicham Toumi, Kamal El Guemmat, Al Habib Benlahmar, Mohamed Talea. A Survey on Routing Protocols for Vehicular Ad-Hoc Networks [Електронний ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/321034805_A_Survey_on_Routing_Protocols_for_Vehicular_Ad-Hoc_Networks
10. Mohammed Ali Hezam, Al Junaid, Syed A. A, Mohd Nazri Mohd Warip, Ku Nurul Fazira Ku Azir, Nurul Hidayah Romli. Classification of Security Attacks in VANET: A Review of Requirements and Perspectives [Електронний ресурс] / Режим доступу до ресурсу: https://www.matec-conferences.org/articles/matecconf/pdf/2018/09/matecconf_mucet2018_06038.pdf
11. Ujwal Parmar, Sharanjit Singh. Overview of Various Attacks in VANET [Електронний ресурс] / Режим доступу до ресурсу: https://pdfs.semanticscholar.org/a5a6/ca0815a6df969a810b03c22b48453fe3236c.pdf
12. Rakesh Shrestha, Rojeena Bajracharya, and Seung Yeob Nam. Challenges of Future VANET and Cloud-Based Approaches [Електронний ресурс] / Режим доступу до ресурсу: https://www.hindawi.com/journals/wcmc/2018/5603518/
13. Manjusha Abhijeet Deshmukh. Challenges in Vehicle Ad Hoc Network (VANET) [Електронний ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/326250676_Challenges_in_Vehicle_Ad_Hoc_Network_VANET
14. Benzekki, Kamal; El Fergougui, Abdeslam; Elbelrhiti Elalaoui, Abdelbaki. Software-defined networking (SDN): A survey [Електронний

ресурс] / Режим доступу до ресурсу:3 https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1737

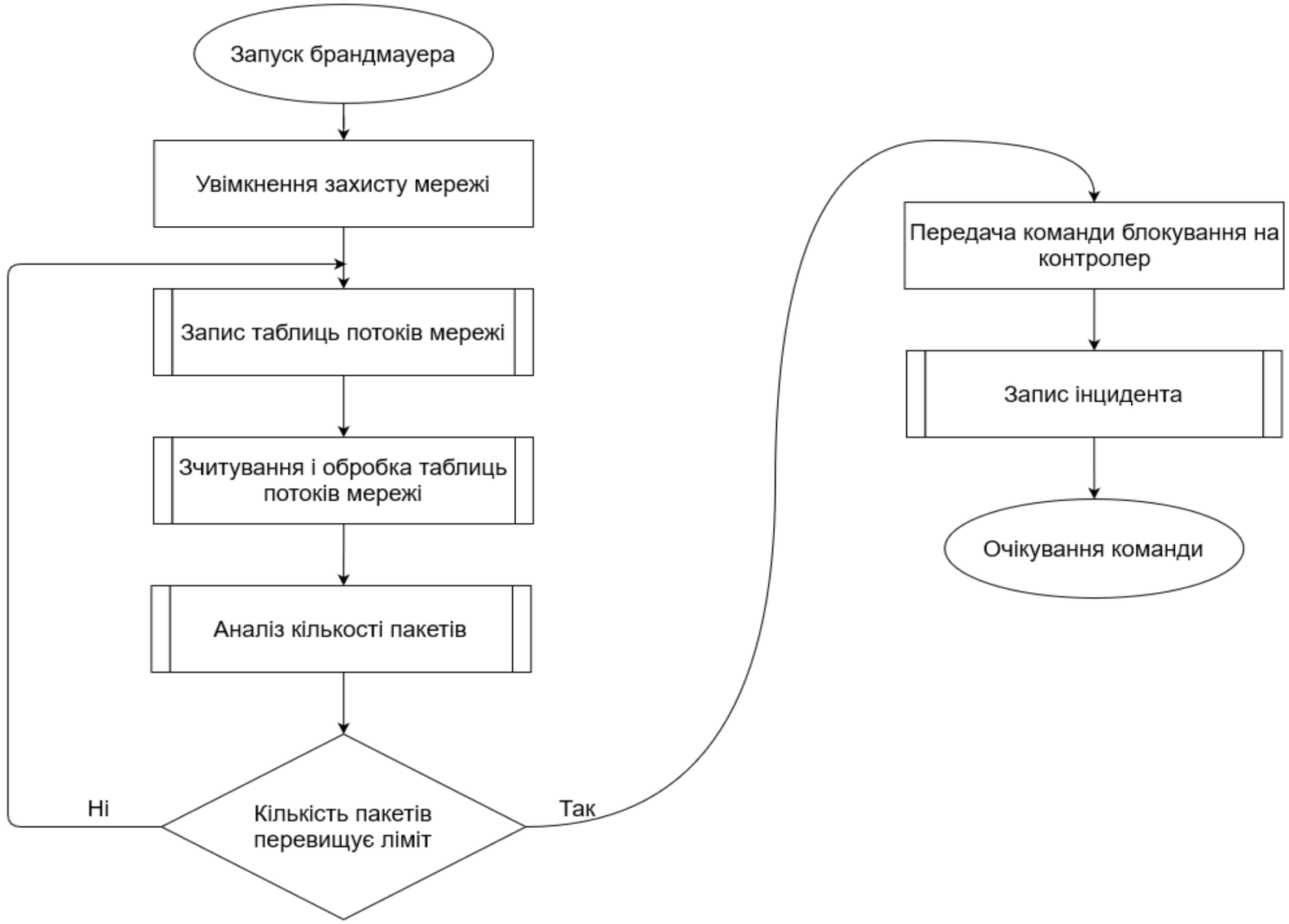
15. Konstantinos Poularakis, George Iosifidis. SDN-Enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge. [Електронний ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/322355141_SDN-Enabled_Tactical_Ad_Hoc_Networks_Extending_Programmable_Control_to_the_Edge
16. Ramon dos Reis Fontes, Christian Esteve Rothenberg. Mininet-Wifi – the User Manual [Електронний ресурс] / Режим доступу до ресурсу: https://usermanual.wiki/Pdf/mininetwifidraftmanual.297704656/view
17. Arvind Rongala. Benefits of Python over Other Programming Languages [Електронний ресурс] / Режим доступу до ресурсу: https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/
18. List of SDN controller software, [Електронний ресурс] / Режим доступу до ресурсу: https://en.wikipedia.org/wiki/List_of_SDN_controller_software
19. Qing Wang, Geddings Barrineau, Ryan Izard. Floodlight Controller. [Електронний ресурс] / Режим доступу до ресурсу: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview?homepageId=1343545
20. DockerHub. Floodlight controller, [Електронний ресурс] / Режим доступу до ресурсу: https://hub.docker.com/r/glefevre/floodlight



					ІАЛЦ.045430.004 Д1		
Зм.	Лист	№ докум.	Підп.	Дата	Принципова схема класів брандмауера	Літера	Маса / Масштаб
Розроб.		Косюга О.Є.					
Перевірив		Калюжний О.О.				Лист 1	Листів 1
Т.контр.							
Н.контр.		Сімоненко В.П.					
Затв.							



					ІАЛЦ.045430.005 Д2	
Зм.	Лист	№ докум.	Підп.	Дата	Структурна схема мережі SDN VANET з брандмауером	Літера / Маса / Масштаб
Розроб.		Косюга О.Є.				
Перевірив		Калюжний О.О.				Лист 1 / Листів 1
Т.контр.						
Н.контр.		Сімоненко В.П.				
Затв.						



					ІАЛЦ.045430.006 Д3			
						Літера	Маса	Масштаб
Зм.	Лист	№ докум.	Підп.	Дата	Функціональна схема алгоритму роботи брандмауеру			
Розроб.		Косюга О.Є.				Лист 1	Листів 1	
Перевірив		Калюжний О.О.						
Т.контр.								
Н.контр.		Сімоненко В.П.						
Затв.								

ДОДАТОК А

Лістинг програми

```
import paramiko as p
import time as t
import re
import os
from tkinter import *
import Tkinter as tk
import threading


class Writer:

    def __init__(self, host, port, user, password, file):
        self.file = file
        self.password = password
        self.user = user
        self.host = host
        self.port = port

    def open_SSH_Session(self):
        client = p.SSHClient()
        client.set_missing_host_key_policy(p.AutoAddPolicy)
        client.connect(hostname=self.host username=self.user,
password=self.password, port=self.port)
        channel = client.invoke_shell()
        print("Session has been opened")
        return channel

    def set_root_channel(self, channel):
```

```
            channel.send('sudo ls' + '\n')
            channel.send(self.password + '\n')

        def send_command(self, command, channel, root=bool):
            if root:
                channel.send('sudo ' + command + '\n')
            else:
                channel.send(command + '\n')

        def write_dumps(self, channel, switches):
            for i in switches:
                self.send_command(command='ovs-ofctl  dump-flows  ' + i + '\n',
channel=channel)
            t.sleep(1)
            out = channel.recv(10000)
            file = open(self.file, 'a')
            file.write(out)

        def clear_dumps(self):
            file = open(self.file, 'w')
            file.write("")


    class Reader:

        def __init__(self, file_location, dump):
            self.file_location = file_location
            self.dump_file_location = dump
            self.lines = []
```

```
            self.flag = True

        def read_dumps(self, num_of_lines):
            self.find_matches(num_of_lines)
            self.filter_results()

        def find_matches(self, num_of_lines):
            file = open(self.file_location, 'r')
            refined_log = open(self.dump_file_location, "a")
            for i in range(num_of_lines):
                line = re.findall(

'(duration=\d+.\d{1,3}).*(n_packets=\d+).*(n_bytes=\d+).*(nw_src=(\d|\.)+),(nw
_dst=(\d|\.)+)',
                    file.readline())
                if line:
                    self.lines.append(list(line[0]))
                    refined_log.write(str(line[0]))


        def filter_results(self):
            for i in self.lines:
                del i[4]
                del i[len(i) - 1]

        def get_results(self):
            return self.lines

        def get_results_string(self):
```

```
        c = 0
        for i in self.lines:
            string = ''
            for x in i:
                string += x + ' '
            c += 1
            print(str(c) + string + '\n')

    def clear_dumps(self):
        self.lines = []


class Analyser:

    def __init__(self):
        self.intruders = set([])

    def find_DDOS(self, dumps):
        txt = ""
        for i in range(0, len(dumps) - 4, 4):
            first_p = int(dumps[i][1][10:len(dumps[i][1])])
            last_p = int(dumps[i + 4][1][10:len(dumps[i + 4][1])])
            intruder_ip = dumps[i][3][7:len(dumps[i][3])]
            if intruder_ip not in self.intruders:
                if last_p - first_p > 500:
                    self.intruders.add(intruder_ip)
                    txt = "Intruder has been caught and banned! IP: " + intruder_ip
                    self.ban_intruder(intruder_ip)
        return txt
```

```
    def get_intruders(self):
        return list(self.intruders)

    def unban_user(self, ip):
        for i in ip:
            if i in self.intruders:
                self.intruders.remove(i)

    def flush_intruders(self):
        self.intruders.clear()

    def ban_intruder(self, ip):
        self.intruders.add(ip)
        os.system(
            'curl -X POST -d \'{\"src-ip\":\"' + ip + "/8" +
            '\","action\":\"deny\"}\"
            ' http://localhost:8080/wm/acl/rules/json')


class Wrapper:

    def __init__(self):
        self.txt = "Welcome to the network manager!\nTo find potential DDOS
intruders, click on the " \
                   "'Activate network protection' button. "
        self.root = Tk()
        self.data = Data()
        self.prot_thread = None
```

```
        self.reader_thread = None
        self.main_thread = None
        self.unban_thread = None
        self.flags = {"console": True, "protect": True, "ban": True}
        self.logs_txt = None
        self.window = None
        self.window_txt = None
        self.main_txt_window = None
        self.users_txt = None
        self.intruders = []

    def create_main_window(self):
        self.root.title("Network Defender by Alexey Kosyuga")
        Button(self.root, text="Show console",
               command=lambda: self.console_button_func()).grid(row=0,
column=0)
        Button(self.root, text='Activate network protection',
               command=lambda: self.start_thread(txt="Starting the network
protection.", func=lambda:
               self.network_protection())).grid(row=0, column=1)
        Button(self.root, text='Stop network protection',
               command=lambda: self.stop_thread(flag=self.flags.get("protect"),
                                                txt="Stopping the network protection.")). \
            grid(row=0, column=2)
        Button(self.root, text="Ban list",
               command=lambda: self.ban_button_func()).grid(row=0, column=3)
        Button(self.root, text="Check logs", command=lambda:
self.check_refined_logs()) \
               .grid(row=0, column=4)
```

```
        Button(self.root, text="Check logs", command=lambda:
self.check_refined_logs())\
            .grid(row=0, column=5)
        Button(self.root, text="Reset rules",
               command=lambda: self.reset_rules()).grid(row=0, column=5)
        Button(self.root, text="Quit",
               command=lambda: self.quit()).grid(row=0, column=6)

    def start_thread(self, txt, func):
        if txt is not None:
            self.txt = txt
        thread = threading.Thread(target=func)
        thread.start()

    def stop_thread(self, flag, txt=None):
        if txt is not None:
            self.txt = txt
        self.flags.update({flag: False})

    def network_protection(self):  # button
        self.data.writer.clear_dumps()
        while self.flags.get("protect"):
            self.data.writer.write_dumps(self.data.channel, self.data.switches)
            self.data.reader.read_dumps(1000)
            self.data.logs += self.data.reader.get_results()
            self.txt = self.data.analyser.find_DDOS(self.data.logs)
            self.data.reader.clear_dumps()

    def check_banned(self):
```

```
            while self.flags.get("ban") and self.logs_txt is not None:
                for i in self.data.analyser.get_intruders():
                    if i not in self.intruders:
                        self.logs_txt.insert(1.0, "Banned user: " + i + "\n")
                    self.intruders.append(i)


        def check_refined_logs(self):
            path = self.data.get_dump_path()
            ref_logs = open(path, "w")
            ref_logs.write("Logs haven't been initiated yet!\n"
                           "Please remember that logs from previous session are automatically deleted.")
            os.system("xdg-open " + self.data.get_dump_path())

        def ban_button_func(self):
            self.window = tk.Toplevel(self.root)
            self.window.title("List of banned users")
            self.logs_txt = Text(self.window)
            self.logs_txt.config(height=20, width=30)
            self.logs_txt.grid(row=3, column=0, columnspan=2, pady=10, padx=10)
            Button(self.window, text="Unban user", command=lambda: self.unban_window_func()).grid(row=0, column=0)
            Button(self.window, text="Ban user", command=lambda: self.ban_window_func()).grid(row=0, column=1)
            self.flags["ban"] = True
            self.start_thread(txt="Opening window with banned users", func=lambda: self.check_banned())
```

```
    def console_button_func(self):
        self.stop_thread(flag="console", txt=None)
        t.sleep(0.5)
        self.flags.update({"console" : True})
        self.start_thread(txt=None, func=lambda: self.console())

    def unban_window_func(self):
        self.window = tk.Toplevel(self.root)
        self.window.title = ("Unbanning users")
        self.users_txt = Text(self.window)
        self.users_txt.config(height=10, width=50)
        self.users_txt.grid(row=0, column=0)
        self.users_txt.insert(1.0,  "Please enter all IP's to unban separated by ',':\n\n")
        Button(self.window, text="Confirm", command=lambda: self.unban()).grid(row=1, column=0)

    def ban_window_func(self):
        self.window = tk.Toplevel(self.root)
        self.window.title = ("Banning users")
        self.users_txt = Text(self.window)
        self.users_txt.config(height=10, width=50)
        self.users_txt.grid(row=0, column=0)
        self.users_txt.insert(1.0, "Please enter all IP's to ban separated by ',':\n\n")
        Button(self.window, text="Confirm", command=lambda: self.ban_func()).grid(row=1, column=0)
```

```
def unban(self):
    usr = str(self.users_txt.get("2.0", 'end-1c'))
    usr = usr.split(",")
    for i in range(0, len(usr)):
        usr[i] = usr[i].strip()
        string = "Banned user: " + usr[i]
        index = self.logs_txt.search(string, 1.0, END)
        self.logs_txt.delete(index, "1." + str(len(index)))
    self.data.analyser.unban_user(usr)
    self.intruders = []
    self.txt = "Following users have been unbanned:\n" + str(usr)
    self.window.destroy()

def ban_func(self):
    usr = str(self.users_txt.get("2.0", 'end-1c'))
    usr = usr.split(",")
    for i in range(0, len(usr)):
        if usr[i] not in self.data.analyser.get_intruders():
            usr[i] = usr[i].strip()
            self.data.analyser.ban_intruder(usr[i])
        # self.logs_txt.insert(END, "Banned user: " + usr[i])
    self.txt = "Following users have been banned:\n" + str(usr)
    self.window.destroy()

def reset_rules(self):
    self.txt = "The controller rules and all restrictions have been eliminated"
    os.system("curl http://localhost:8080/wm/acl/clear/json")
    self.data.writer.clear_dumps()
    self.data.reader.clear_dumps()
```

```
        self.data.analyser.flush_intruders()
        self.data.logs = []
        self.intruders = []
        self.data.writer.clear_dumps()
        if self.logs_txt:
            self.logs_txt.delete(1.0, END)

    def start_loop(self):
        self.create_main_window()
        self.root.mainloop()

    def console(self):
        prev_text = None
        if self.main_txt_window:
            prev_text = self.main_txt_window.get(1.0, END)
        self.main_txt_window = Text(self.root)
        self.main_txt_window.config(height=20, width=100)
        self.main_txt_window.grid(row=1, column=0, columnspan=7, pady=10)
        if self.main_txt_window and prev_text:
            self.main_txt_window.insert(1.0, prev_text)
        c = 1
        while self.flags.get("console"):
            if self.txt:
                txt = "\n" + str(c) + ". " + self.txt
                self.main_txt_window.insert(END, txt)
                self.txt = None
                c += 1
```

```
    def quit(self):
        self.data.analyser.flush_intruders()
        self.stop_thread(flag="console", txt=None)
        self.stop_thread(flag="protect", txt=None)
        self.stop_thread(flag="ban", txt=None)
        self.root.destroy()


class Data:

    def __init__(self):
        self.host = '192.168.57.4'
        self.port = 22
        self.user = 'wifi'
        self.password = 'wifi'
        self.refined_logs_path = "/home/ambel/Documents/REFINED_LOGS"
        self.logs_path = "/home/ambel/Documents/SWITCH_LOGS"
        self.reader = Reader(self.logs_path, self.refined_logs_path)
        self.analyser = Analyser()
        self.switches = ['ap4', 'ap1', 's1']
        self.writer = Writer(host=self.host, port=self.port,
password=self.password, user=self.user, file=self.logs_path)
        self.channel = self.writer.open_SSH_Session()
        self.writer.set_root_channel(self.channel)
        self.intruders = []
        self.logs = []

    def get_dump_path(self):
        return self.refined_logs_path
```

```
wrapper = Wrapper()
wrapper.start_loop()
```