Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему **МАСШТАБНА ДЕЦЕНТРАЛІЗОВАНА MESH МЕРЕЖА НА ОСНОВІ YGGDRASIL**

Виконав: студент 2 курсу, групи КНТ-513м
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні системи та мережі

ПЕСТОВ О. Д.
(ПРІЗВИЩЕ та ініціали)
Керівник КИРИЧЕК Г. Г.
(ПРІЗВИЩЕ та ініціали)
Рецензент МАЛИЙ О. Ю.
(ПРІЗВИЩЕ та ініціали)

2024

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) «Комп'ютерні системи та мережі»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Зав. кафедри Кудерметов Р.К.
_____
"____" _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПЕСТОВА Олексія Дмитровича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Масштабна децентралізована mesh мережа на основі Yggdrasil
керівник проєкту (роботи) к. т. н., доцент, КИРИЧЕК Галина Григорівна
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від "18" жовтня 2024 року № 149
2. Строк подання студентом проєкту (роботи) 1 грудня 2024 року
3. Вихідні дані до проєкту (роботи) наявні схеми та протоколи маршрутизації mesh мереж, експериментальна схема маршрутизації Yggdrasil, середовища meshnet-lab та coreemu-lab, операційна система OpenWRT
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз предметної області, дослідження наявних схем маршрутизації mesh мереж;
2) Дослідження схеми маршрутизації Yggdrasil;
3) Розгортання мережі;
4) Реалізація прототипу вузла мережі.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Принципова електрична схема прототипу вузла мережі

6. Консультанти розділів проєкту (роботи**)**

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	КИРИЧЕК Г. Г., доцент		
нормоконтроль	ЩЕРБАК Н. В., ст. викл.		

7. Дата видачі завдання 23.09.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області, дослідження наявних	30.09.2024 р.	
	схем маршрутизації mesh мереж		
2	Дослідження схеми маршрутизації Yggdrasil	27.10.2024 р.	
3	Визначення технологій, програмного та апаратного	10.11.2024 р.	
	забезпечення, необхідних для розгортання мережі		
4	Реалізація прототипу вузла мережі	20.11.2024 р.	
5	Тестування розробленого пристрою	23.11.2024 р.	
6	Оформлення отриманих результатів у ПЗ	01.12.2024 р.	

Студент _____ **Олексій ПЕСТОВ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Галина КИРИЧЕК**
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 81 с; 48 рис; 3 табл; 23 джерела.

МАСШТАБОВАНІСТЬ, MESH МЕРЕЖА, YGGDRASIL, MESHNET-LAB, COREEMU-LAB, WI-FI, 802.11s, OPENWRT

Метою даної дипломної роботи є проведення досліджень, визначення доцільності використання схеми маршрутизації Yggdrasil, як основи масштабної децентралізованої mesh мережі, а також проведення реалізації і тестування прототипу вузла мережі, яка проєктується.

Об’єктом дослідження є процес проєктування масштабної децентралізованої mesh мережі на основі схеми маршрутизації Yggdrasil.

Предметом дослідженню є моделі, методи та програмні засоби реалізації масштабної децентралізованої mesh мережі на основі Yggdrasil.

В роботі досліджується масштабованість експериментальної схеми маршрутизації Yggdrasil щодо логіки, ліміту переходів, використання процесора та пам’яті. Експерименти з демоном маршрутизації проведено на різного розміру топологіях із використанням середовищ meshnet-lab і coreemu-lab. Виявлено певні особливості використання даної схеми маршрутизації як основи масштабної децентралізованої mesh мережі.

Для mesh мережі на основі Yggdrasil визначено базову технологію канального рівня. Реалізовано прототип вузла мережі на основі ОС OpenWRT, описано процес прошивки і налаштування та його автоматизацію. Прототипи протестовано, визначено орієнтовні ліміт покриття окремого вузла та пропускну здатність мережі при даній конфігурації.

ABSTRACT

Explanatory note to the master's work: 81 p.; 48 figures; 3 tables; 23 sources.

SCALABILITY, MESH NETWORK, YGGDRASIL, MESHNET-LAB, COREEMU-LAB, WI-FI, 802.11S, OPENWRT

The purpose of this thesis is to conduct research, determine the feasibility of using the Yggdrasil routing scheme as a basis for a large-scale decentralized mesh network, as well as to implement and test a prototype node of said network.

The object of the research is the models, methods and software tools for implementing a large-scale decentralized mesh network based on Yggdrasil.

This work investigates the scalability of the experimental Yggdrasil routing scheme in terms of logic, hop limit, CPU and memory usage. Experiments with the routing daemon were conducted on topologies of different sizes using meshnet-lab and coreemu-lab environments. Certain caveats of using this routing scheme as the basis for a large-scale decentralized mesh network have been revealed.

For an Yggdrasil-based mesh network the preferred link-layer technology was determined. A prototype of a network node based on the OpenWRT OS was developed, the process of firmware flashing and configuration, as well as its automation was described. The developed prototypes were tested. The approximate coverage limit of a single node and the network bandwidth with this configuration were determined.

ЗМІСТ

ВСТУП

В наші дні централізовані програмно-визначені мережі (Software-Defined Networking, SDN) стають все більш популярними серед інтернет-провайдерів (ISP) завдяки перевагам у автоматизованому віддаленому налаштуванні обладнання, моніторингу, контролі, простоті розгортання, використанні апаратного забезпечення типу «white box» тощо [1]. Тим не менш, згадані переваги мають місце за рахунок надійності мережі та стійкості до збоїв, оскільки централізація неминуче додає системі єдині точки відмови. Наслідком цього є, наприклад, відносно недавній сумнозвісний колапс мережі Київстар 12 грудня 2023 року, що призвів до кількох днів простою з повною відсутністю зв’язку у клієнтів. Численні сервіси по всій Україні покладалися на «Київстар» і не могли функціонувати, поки мережу не було відновлено.

Це є прикладом того, наскільки ненадійними є централізовані ієрархічні системи зв'язку та вимагає іншого підходу - сітчастої (mesh) мережі, де межа між маршрутизаторами та кінцевими пристроями стає розмитою. Ідея цієї парадигми полягає в тому, що кожен вузол мережі функціонує незалежно від будь-якого іншого, встановлюючи зв'язки з іншими вузлами поблизу (peering) і співпрацюючи з метою маршрутизації трафіку для учасників мережі, а не покладаючись для цього на центральний орган.

Хоча концепція mesh мереж не нова, ряд протоколів маршрутизації вже розроблено (OLSR, Babel, B.A.T.M.A.N. тощо) і успішно використано (ініціатива Freifunk та інші громадські мережі [2]), нема протоколу, який би масштабувався за межі невеликих локальних мереж і робив це ефективно. Додатковою проблемою таких мереж є підтримка конфіденційності даних користувача при передачі їх через мережу з урахуванням того, що вони можуть прослуховуватись проміжними вузлами [3].

Цей дипломний проект присвячено проведенню досліджень з теоретичних та практичних питань розгортання масштабної децентралізованої mesh мережі. Об’єктом дослідження є процес проєктування масштабної децентралізованої mesh мережі на основі схеми маршрутизації Yggdrasil. Предметом є моделі, методи та програмні засоби реалізації масштабної децентралізованої mesh мережі на основі Yggdrasil.

У першому розділі розглянуто суміжні дослідження та роботи. Наведено загальний опис функціоналу та особливостей схеми маршрутизації Yggdrasil у порівнянні з іншими схемами та протоколами маршрутизації у mesh мережах. Визначено завдання, які виконуються у ході роботи.

Другий розділ присвячено дослідженню Yggdrasil з приводу її масштабованості відносно ліміту переходів, використання системних ресурсів та логіки роботи. Проведено низку експериментів як у великому, так і у малому масштабах, зроблено висновки щодо доцільності використання схеми маршрутизації для побудови масштабних mesh мереж.

У третьому розділі визначено бажану базову технологію зв’язку канального рівня, наведено орієнтовну структуру проєктованої мережі. Визначено програмне та апаратне забезпечення необхідне для реалізації прототипу вузла mesh мережі на основі Yggdrasil.

Четвертий розділ присвячено процесу реалізації прототипу вузла мережі. Наведено принципіальну електричну схему пристрою, розглянуто процес збірки та прошивки пристрою. Розроблено скрипти автоматизації процесу прошивки та первісного налаштування пристрою. Проведено перевірку загальної працездатності пристрою та мережі, визначено орієнтовний радіус покриття окремого вузлу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Суміжні дослідження та роботи

Більшість досліджень у сфері маршрутизації для великих mesh мереж зосереджені на її реалізації для IoT та бездротових сенсорних мереж, зокрема із фокусуванням на оптимізації енергоспоживання і віддаленості за рахунок пропускної здатності та безпеки. Наприклад, мережа на основі CottonCandy працює за принципом організації вузлів в остовне дерево (spanning tree) від вузла Інтернет-шлюзу, призначеного коренем дерева, при цьому уникаючи зіткнень завдяки використанню рекурсивних запитів даних під час робочих циклів (рис 1.1) [4].

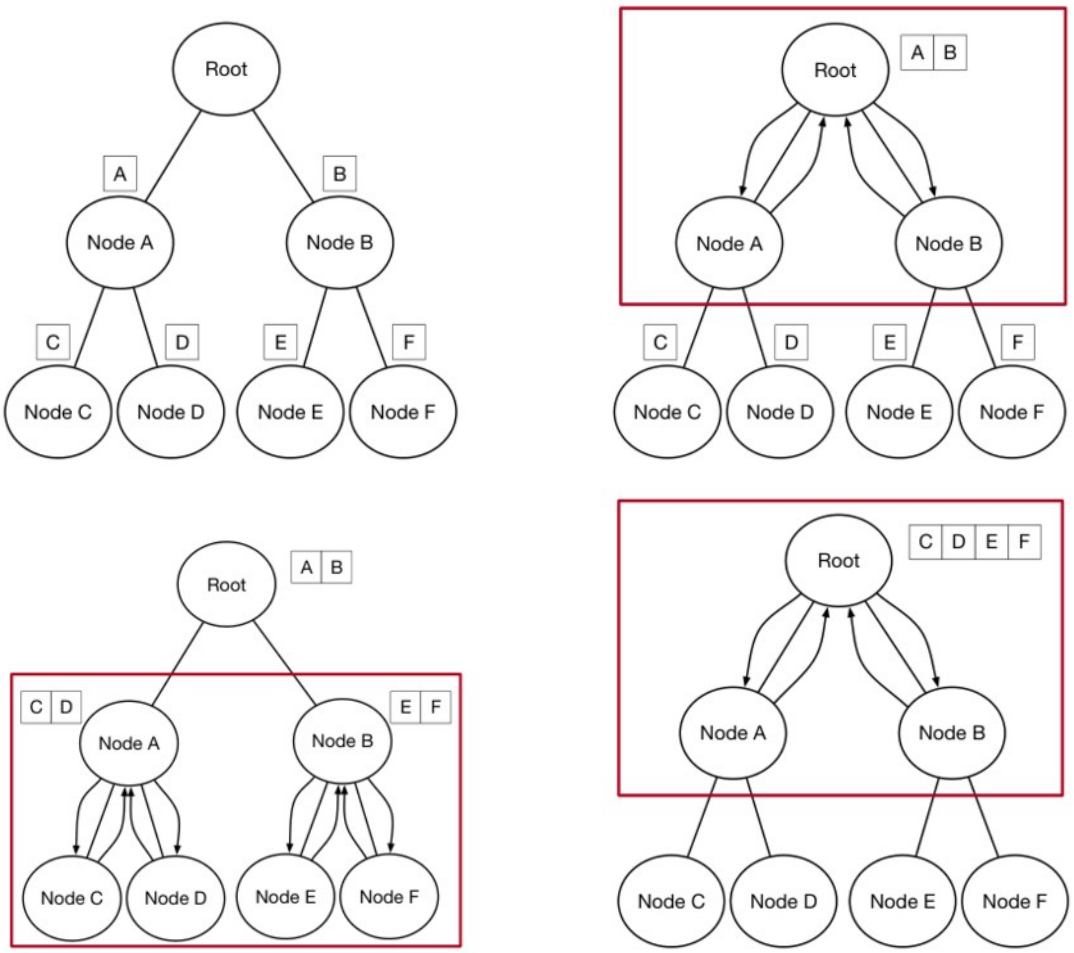


Рисунок 1.1 - Рекурсивне збирання даних у остовному дереві CottonCandy [4]

Додатковим засобом уникнення зіткнень є використання вузлами різних каналів відповідно до остовного дерева — кожен вузол прослуховує батьківський канал на предмет рекурсивних запитів, а пропагує їх за власним каналом, відмінним від батьківського (рис. 1.2).

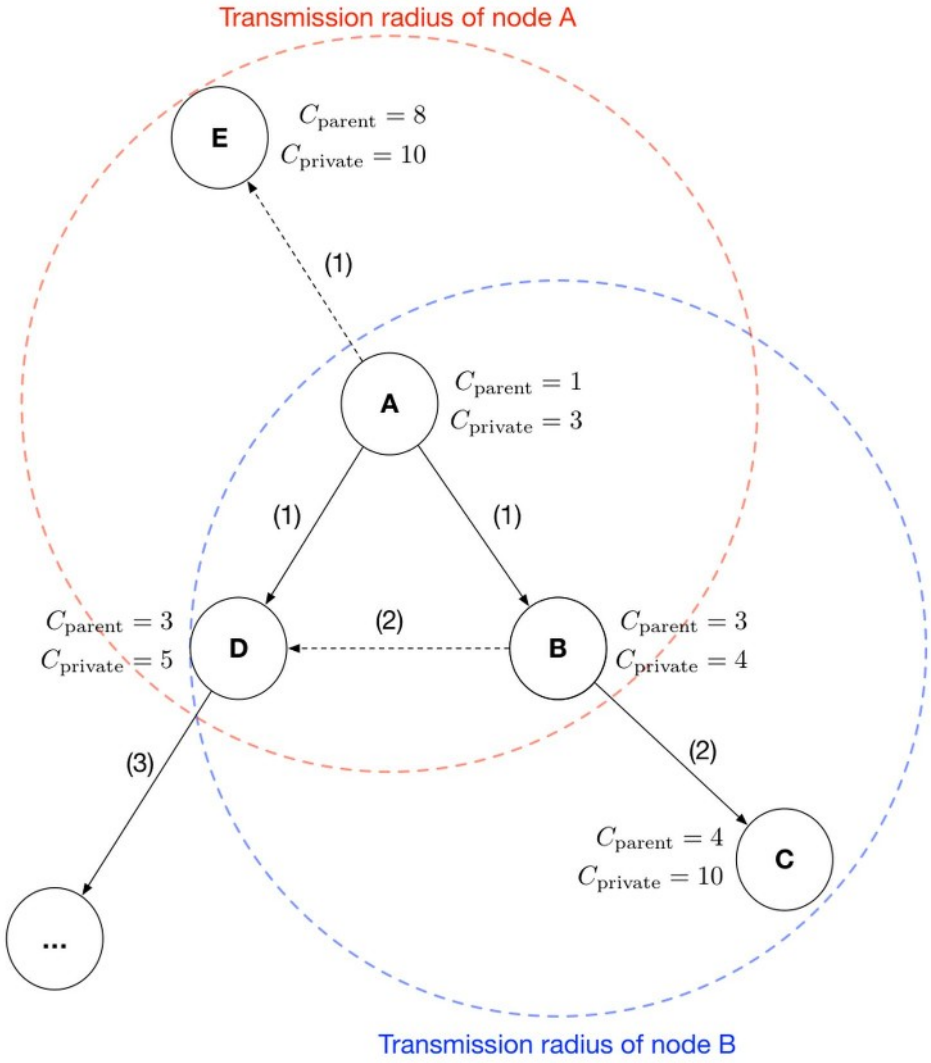


Рисунок 1.2 - Канальний розподіл CottonCandy [4]

Хоча цей підхід добре працює при невеликих обсягах сенсорних даних, що періодично збираються, він не підходить для спонтанного зв’язку між вузлами в мережі загального призначення. Крім того, наявність чітко визначеного кореневого вузла додає до мережі єдину точку збою, що перешкоджає загальній меті децентралізації.

Ямамото та інші автори пропонують протокол маршрутизації VORTEX - підхід, заснований на опортуністичній маршрутизації ієрархічної структури рівнів, яка встановлена на початковому етапі [5] (рис. 1.3). Рівні розподіляються за наступним алгоритмом:

- вузли періодично надсилають HELLO-пакети та прослуховують ефір на предмет цих пакетів від інших вузлів;

- з кількості унікальних ID у отримуваних HELLO-пакетах кожен вузол може дізнатись кількість своїх сусідів (ступінь) та їх ступені;

- якщо даний вузол має найбільшу ступінь з сусідів, він самостійно підвищує свій ступінь до 1, інакше надсилає UTR (upper tier request, запит підвищення рівня) вузлу з найбільшим ступенем;

- отримувач UTR підвищує свій рівень до 1;

- аналогічно серед вузлів першого рівня обираються вузли другого рівня.

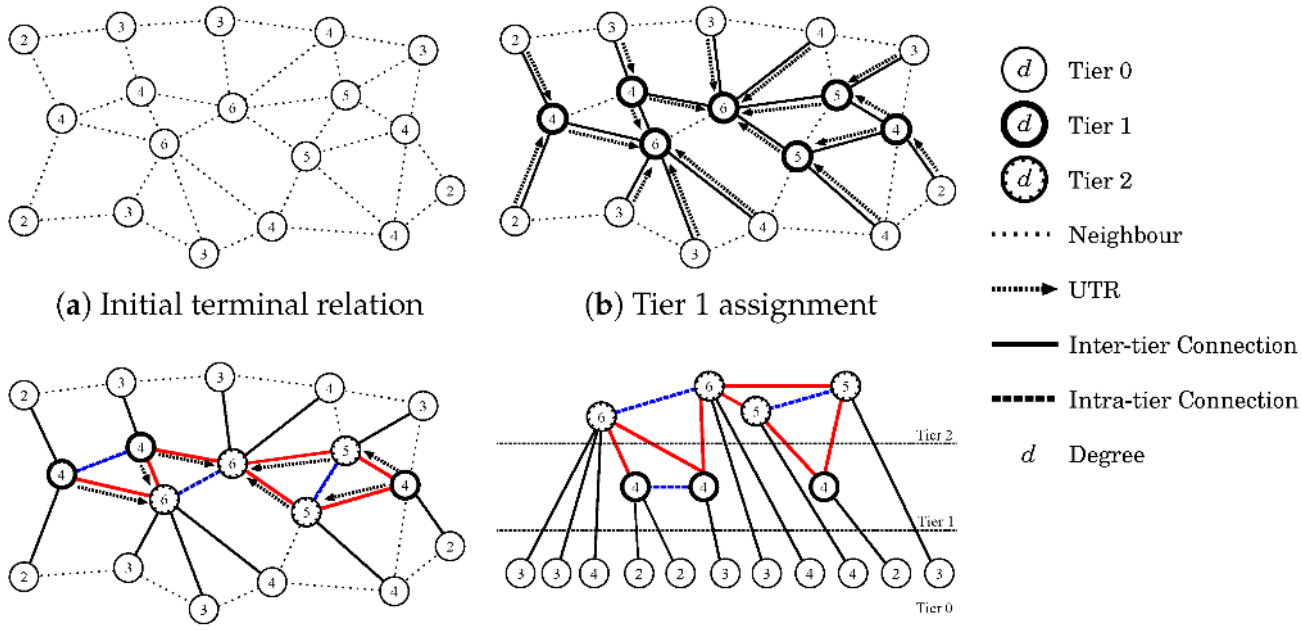


Рисунок 1.3 - Розподіл вузлів за рівнями у VORTEX [5]

Для пари кінцевих вузлів, що спілкуються, мережа виглядає як «чорна скринька», а проміжні вузли просто пересилають пакети вздовж ієрархії, поки вони

не досягнуть одержувача (рис. 1.4). Пакети проходять кількома шляхами для забезпечення кращої надійності. Таким чином, необхідність здійснювати спочатку виявлення маршруту відпадає, однак мережа переповнюється додатковим трафіком, який, у великих масштабах, значно знижує її продуктивність.

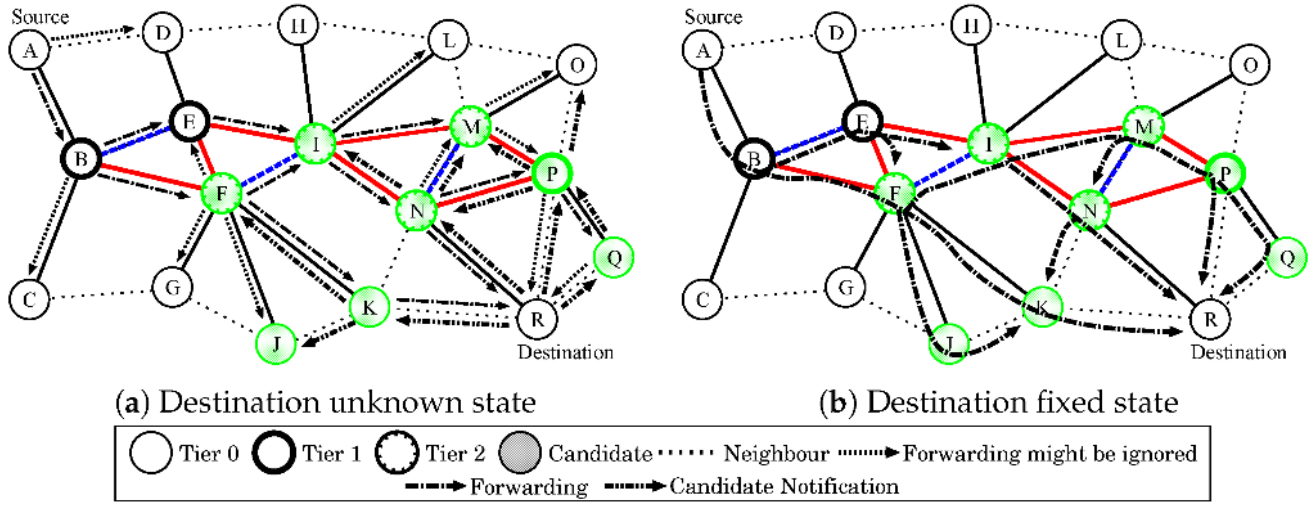


Рисунок 1.4 - Маршрутизація у VORTEX [5]

Прабу та інші [6] представляють новий підхід до mesh маршрутизації, використовуючи концепцію «теплових слідів» і представляючи її за допомогою bloom-фільтрів [7] на кількох рівнях градієнта інтенсивності (рис. 1.5). Кожен вузол підтримує bloom-фільтри декількох рівнів, у які заносить вузли, що знаходяться через 1, 2, 3 переходи і т.д. Таким чином, в кожного вузла наявна інформація про «тепло» інших вузлів поблизу та його інтенсивність.

Під час пошуку маршруту вузли випадковим чином «переглядають» мережу, поки не досягнуть «тепла» вузла призначення у чиємусь bloom-фільтрі останнього рівня, а потім слідують за ним у напрямку збільшення інтенсивності (рис 1.6). Коли пункт призначення досягається, вздовж знайденого шляху формується «тепловий слід» (рис. 1.7). Таким чином, інші вузли, які шукають два попередні, можуть натрапити на нього та прослідувати до місця призначення, повторно

використовуючи маршрут. При цьому знайдений маршрут поступово оптимізується під час використання.

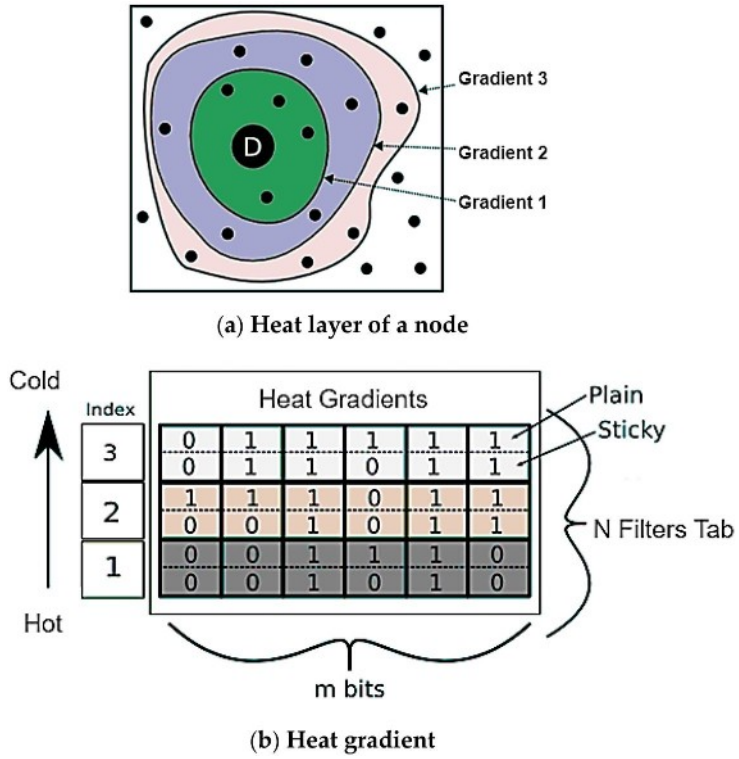


Рисунок 1.5 — Вираження градієнта інтенсивності «тепла» вузла D bloom-фільтрами [6]

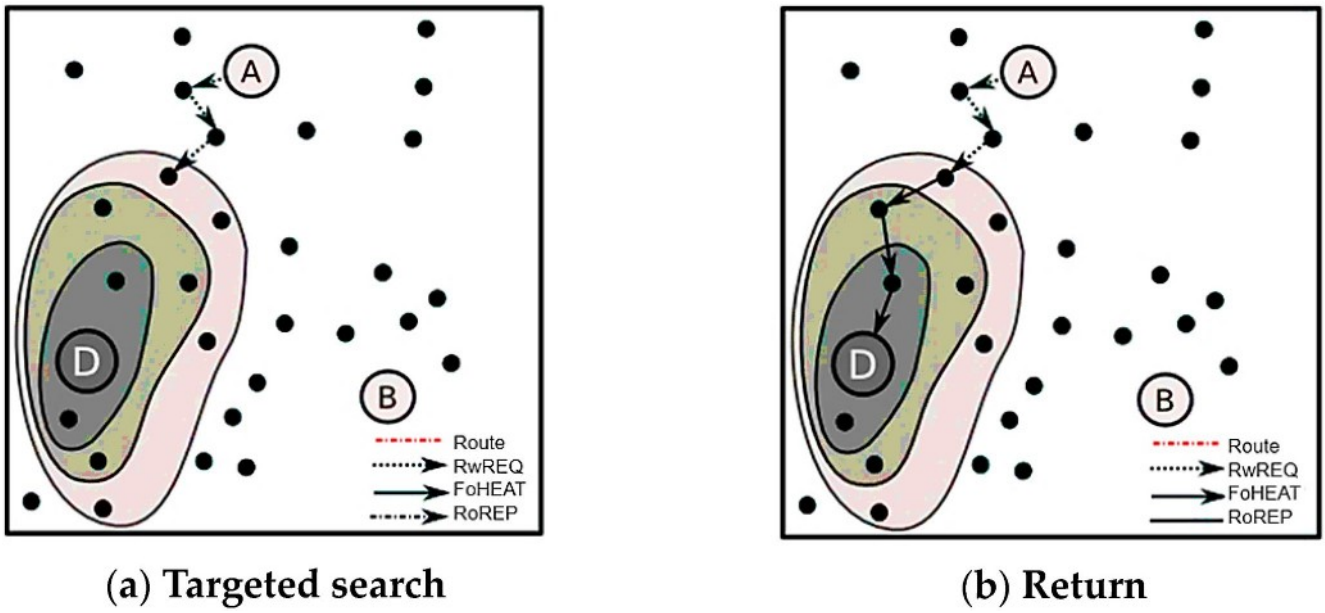


Рисунок 1.6 — Пошук вузла D за його «теплом» [6]

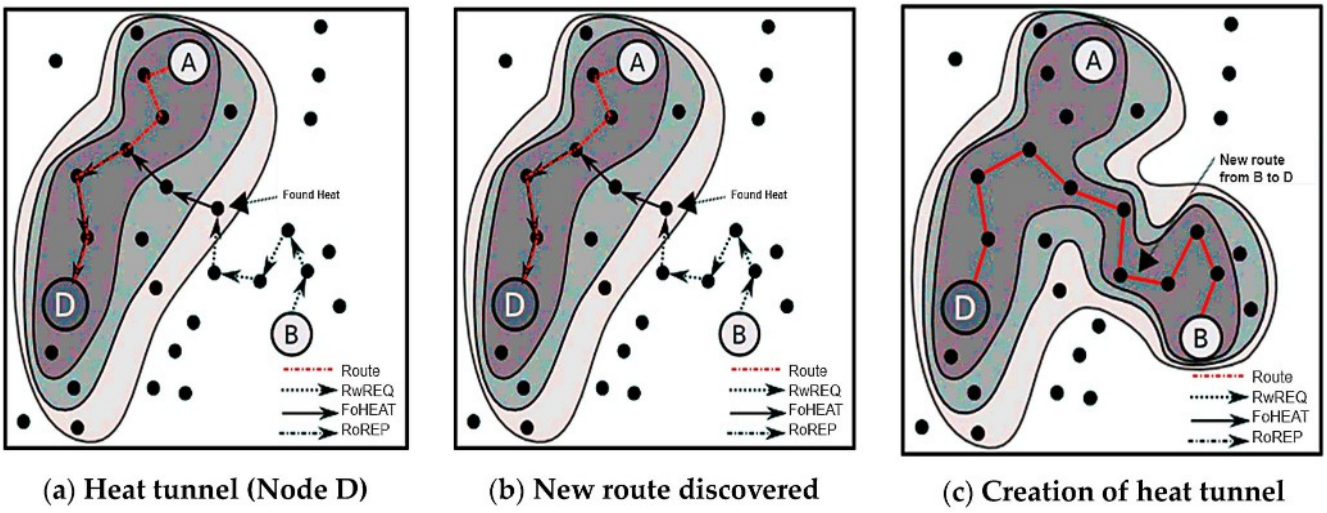


Рисунок 1.7 — Повторне використання «теплового тунелю» вузлом B [6]

Основним недоліком цього підходу є його великомасштабна продуктивність, пов’язана з випадковим пошуком із малою кількістю рівнів градієнта «тепла». Тому іноді, для того, щоб знайти пункт призначення, може знадобитися пройти всю мережу, а збільшення кількості рівнів призведе до збільшеного використання пам'яті.

1.2 Схема маршрутизації Yggdrasil

Потенційним вирішенням вищезазначених проблем є використання експериментальної схеми маршрутизації Yggdrasil [8, 9, 10] (поточна остання версія - v0.5.5), основна суть якої полягає у використанні адресації на основі відкритих ключів Ed25519 [11] і самоорганізованого остовного дерева мережі. Останнє схоже на CottonCandy, за винятком того, що замість визначеного шлюзу коренем дерева стає вузол із найнижчим значенням ключа, і тому він може бути автоматично заміненим у разі збою. Bloom-фільтри присутні на кожній гілці у дереві (on-tree links) та представляють собою набір ключів вузлів, доступних через конкретне з’єднання. Пошук вузлів виконується на вимогу та за допомогою

широкомовних розсилок, які фільтруються цими bloom-фільтрами. Результатом такого пошуку є отримання координат вузла, який приймає дані, відносно поточного кореня мережі, за якими надсилається трафік. Для того, щоб визначити напрямок трафіку, на кожному наступному вузлі застосовується опортуністична жадібна маршрутизація [12]. Іншими словами, пакети надсилаються через будь-яке єдине з'єднання, яке наблизить їх до вказаних координат. Ця частина процесу схожа на роботу VORTEX, бо виявлення маршруту пропускається, а замість цього дані пересилаються спираючись на локальні рішення та на основі ієрархічної структури, але не обов'язково суворо за нею.

Цей спосіб має ряд додаткових переваг:

- вузли самостійно генерують собі адреси;

- використовується прозоре асиметричне наскрізне шифрування;

- завдяки використанню криптографічних ключів для адресації є можливість перевірити джерело/одержувача - вони забезпечують підтвердження особи, дозволяючи обмежувати доступ на основі адреси джерела тощо.

Взаємне виявлення вузлів виконується multicast-розсилкою «маяків» з IPv6 link-local адрес мережевих інтерфейсів. Сусідські TCP-з'єднання між вузлами встановлюються таким же чином, або через будь-які інші явно прописані з'єднання. У пристроях з підтримкою IPv6 link-local адреси для мережевих інтерфейсів генеруються автоматично (зазвичай з MAC-адрес адаптерів), тому для автоматичного пірингу ніякого додаткового налаштування не потрібно. Опціонально для деяких інтерфейсів автоматичне взаємне виявлення сусідів можна відключити.

Перевантаженість мережі (network congestion) автоматично контролюється на кожному вузлі за допомогою форми чесної черги пакетів, яка намагається рівномірно збалансувати потоки трафіку між різними вузлами, де це можливо. Основним недоліком підходу Yggdrasil є те, що трафік не завжди проходить найкоротшим можливим шляхом, створюючи більшу затримку, що можна враховувати і вважати справедливим компромісом. Якість зв'язку також не

розглядається поза пріоритетами між декількома пірингами до одного вузла (наприклад, встановлення пріоритету кабельних інтерфейсів над бездротовими).

Кожен вузол мережі зберігає в пам’яті лише bloom-фільтри у 1 КБ для кожної зі своїх гілок, а також 32-байтні ключі своїх прямих сусідів та їхніх попередників на шляху до поточного кореня. Це означає, що використання пам’яті певним вузлом має залежати лише від кількості з’єднань і відстані до поточного кореня. Що стосується використання процесору, то очікуються значні стрибки на кінцевих передавальних та приймальних вузлах через шифрування і дешифрування відповідно, а також позитивна кореляція із кількістю з’єднань.

Через те, що схема маршрутизації Yggdrasil є відносно новою та все ще перебуває на альфа-стадії розробки, існує не так багато робіт, які охоплюють її функціональність або просто наводять, як приклад. Автори наукових робіт представляють результати застосування порівняльних тестів при використанні оверлейної тестової мережі Yggdrasil як інструменту обходу NAT при віддаленому керуванні, але без тестування масштабованості самої схеми маршрутизації [13]. Інші автори порівнюють Yggdrasil із іншими мережними протоколами маршрутизації в ряді бенчмарків, де лише один з них підтримує масштабування понад 100 вузлів [14]. Варто також зазначити, що у бенчмарку доступність вузла вимірюється лише один раз із обмеженим терміном надходження пакетів. Цей підхід не зовсім підходить для Yggdrasil і маршрутизації на вимогу в цілому, оскільки перший пакет завжди має значно довшу затримку, ніж наступні, через те, що спочатку відбувається пошук вузла.

1.3 Постановка завдань

У ході дослідження необхідно визначити масштабованість експериментальної схеми маршрутизації Yggdrasil стосовно ліміту переходів,

використання оперативної пам’яті та процесору, а саме їх залежності від розмірів мережі, її топології та інших факторів. З цього робиться висновок про доцільність використання схеми маршрутизації Yggdrasil у якості основи для побудову масштабних децентралізованих mesh мереж.

При позитивному висновку дослідження масштабованості та доцільності використання схеми маршрутизації необхідно визначити орієнтовну структуру впроваджуваної mesh мережі та реалізувати прототип вузла, як апаратно-програмну платформу для розгортання mesh мережі на основі Yggdrasil. В це входить визначення бажаної базової технології зв’язку канального рівня, операційної системи та необхідного апаратного забезпечення. Передбачається розробка принципіальної електричної схеми пристрою та скриптів автоматизації процесу прошивки, перевірка загальної працездатності прототипу та мережі, визначення орієнтовного покриття окремого вузла (максимальної дальності між сусідніми вузлами).

2 ДОСЛІДЖЕННЯ СХЕМИ МАРШРУТИЗАЦІЇ YGGDRASIL

2.1 Великомасштабне тестування

2.1.1 Методи

Метою великомасштабного бенчмаркінгу є визначення продуктивності Yggdrasil у великих мережах (50 і більше вузлів) із одночасним записом використання системних ресурсів (процесору і пам’яті) демоном маршрутизації кожного вузла. Таке тестування надає інформацію про технічні специфікації апаратного забезпечення, необхідного для підтримки мережі Yggdrasil, а також визначає ліміт переходів (хопів), якщо він є. При прийнятті рішення про доречність схеми маршрутизації також важливі масштабування, використання системних ресурсів і зростання часу конвергенції із розміром мережі.

Випробування проводилися із використанням середовища meshnet-lab, враховуючи його гнучкість та можливості емуляції великомасштабних топологій за допомогою мережевих просторів імен Linux (Linux network namespaces) [15] та автоматизації процесу за допомогою Python. Однак, слід зазначити, що хост-система, на якій проводилися тести, не може обробляти більше ніж 750 вузлів одночасно, оскільки після цього в системі закінчується пам’ять і починають використовуватися додаткові ресурси.

Далі наведемо процедуру бенчмаркінгу. Вона є покроковою.

Крок 1. Підготовка топології попередньо згенерованими файлами JSON, включеними до meshnet-lab та обмеження пропускної здатності з’єднань до 100 Мбіт/с.

Крок 2. Запуск демонів маршрутизації на вузлах.

Крок 3. Запуск скрипту моніторингу системних ресурсів (кожну секунду скриптом здійснюється запис використання ЦП та резидентної пам’яті).

Крок 4. Очікування 30 секунд, поки вузли знаходять один одного.

Крок 5. Протягом наступних 30 секунд надсилаються N унікальних випадкових пінгів із мінімальною кількістю переходів 2 і дедлайном у 30 секунд, де N є кількістю з’єднань у топології.

Крок 6. Очікування 15 секунд.

Крок 7. Повторюються кроки з 5 по 6 ще п'ять разів.

Крок 8. Запуск сценарію збору інформації про вузли (запитує специфічну для Yggdrasil інформацію, відкриті ключі та представлення дерева від демонів маршрутизації, яка необхідна для визначення кореневого вузла під час обробки результатів).

Крок 9. Зупинка демонів маршрутизації та сценаріїв моніторингу ресурсів і очищення мережі.

Крок 10. Повторення процедури на більшій топології, доки не надійде менше 40% пінгів або не буде застосовано більше 750 вузлів.

Цей алгоритм, як більш реалістичний і найгірший сценарій відповідно, перевірено на топології випадкового дерева і лінійній топології (рис. 2.1). У великих топологіях, щоб мережа досягла конвергенції, потрібно на сьомому кроці збільшити кількість ітерацій. Крім того, з метою дослідження обсягу пам’яті, що використовується в конвергентній мережі, який, як очікується, буде приблизно постійним, проведено тривалі за часом експерименти на найбільшій життєздатній топології. Топологія вважається життєздатною, якщо вона досягає конвергенції, що визначається 100% надходженням пінгів.

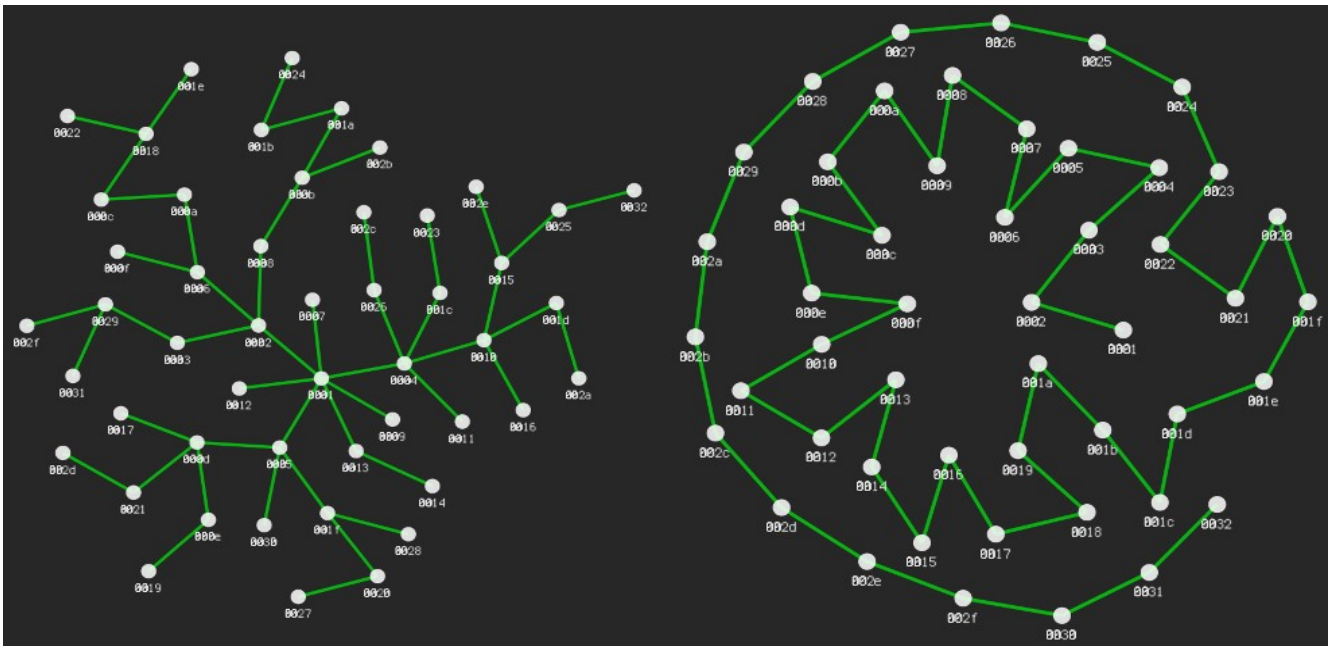


Рисунок 2.1 - Топології випадкового дерева та лінійна на 50 вузлів

Інший експеримент на випадковій деревоподібній топології з 50 вузлів без обмеження пропускної здатності показав різницю у використанні процесору для вузлів, які передають та приймають дані, а також вузлів, які не беруть участі в передачі. Трафік генерувався протягом 10 секунд між двома віддаленими вузлами (наприклад, 001e і 0032, рис. 2.1) за допомогою утиліти iperf3. Нарешті, проведення експерименту на лінійній топології з 750 вузлами із недосконалими повільними з’єднаннями показав стійкість Yggdrasil до таких умов. Застосовано наступні параметри з’єднань: пропускна здатність дорівнювала 10 Мбіт/с із затримкою 10±5

мс. Трафік генерувався між першим і останнім вузлом за допомогою стандартної утиліти ping. Останні два тести проводилися вручну. Процедуру бенчмаркінгу автоматизовано за допомогою модифікованої версії скрипту Python «benchmark1», включеного в meshnet-lab. Отримані дані проаналізовано та відображено графічно за допомогою Python бібліотек pandas, seaborn та Matplotlib [16, 17].

2.1.2 Результати

Результати порівняльного аналізу пам’яті за топологією випадкового дерева представлені на рисунках 2.2 і 2.3. Рисунок 2.2 підтверджує припущення про те, що на використання пам’яті вузлом позитивно впливає кількість його з’єднань (позначено розміром та відтінком точок, а хрестами позначено кореневі вузли). Рисунок 2.3 демонструє, що незважаючи на збільшення загального розміру мережі, використання пам’яті для абсолютної більшості вузлів залишається приблизно у тому ж діапазоні. Надходження пакетів для топології випадкового дерева залишається весь час на рівні 100%.

На рисунках 2.4 і 2.5 наведено результати того ж самого бенчмарку, проведеного на лінійній топології. Оскільки в лінійній топології кожен вузол, крім першого та останнього, має лише два зв’язки, відтінок і розмір точок тут представляють положення вузлів у лінії. Порівняння даних рисунків 2.3 і 2.5 підтверджує, що довжина мережі позитивно впливає на використання пам’яті, що також підтверджується піковим її використанням, наведеним у таблиці 2.1. Ця тенденція набагато більш помітна на лінійних топологіях, хоча для мереж із понад 300 вузлів вона перестає бути настільки постійною.

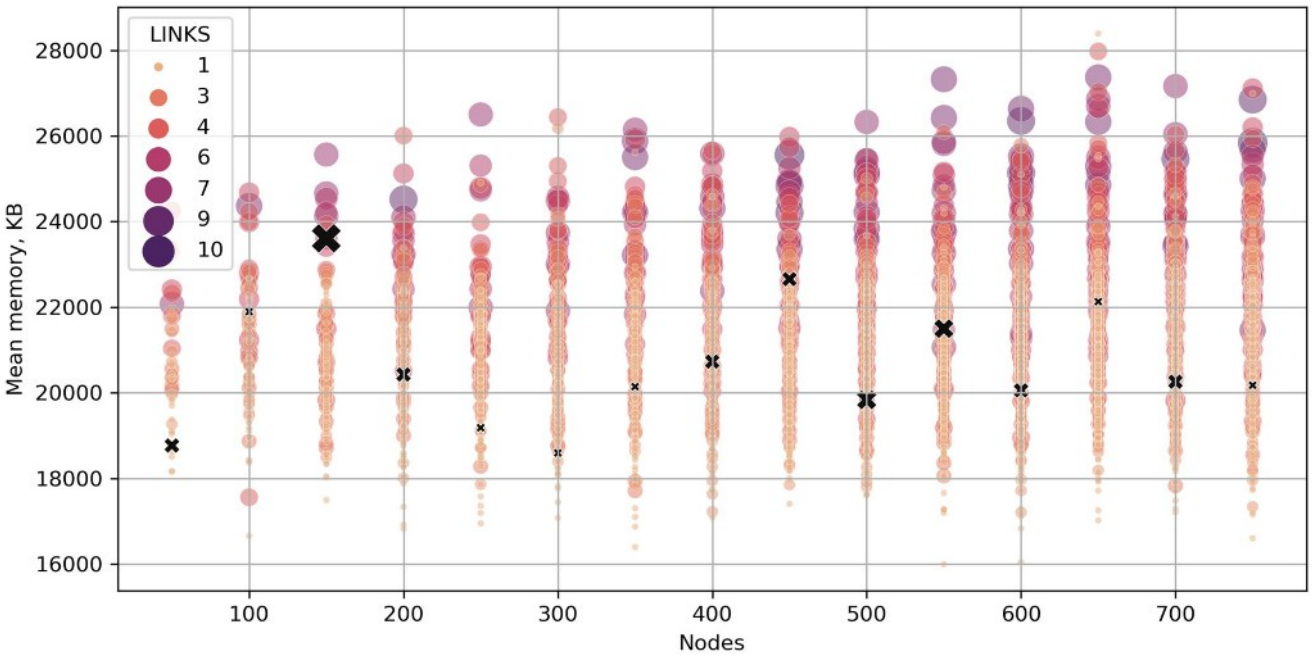


Рисунок 2.2 - Середнє використання пам’яті кожного вузла у мережах різного розміру, топологія випадкового дерева,

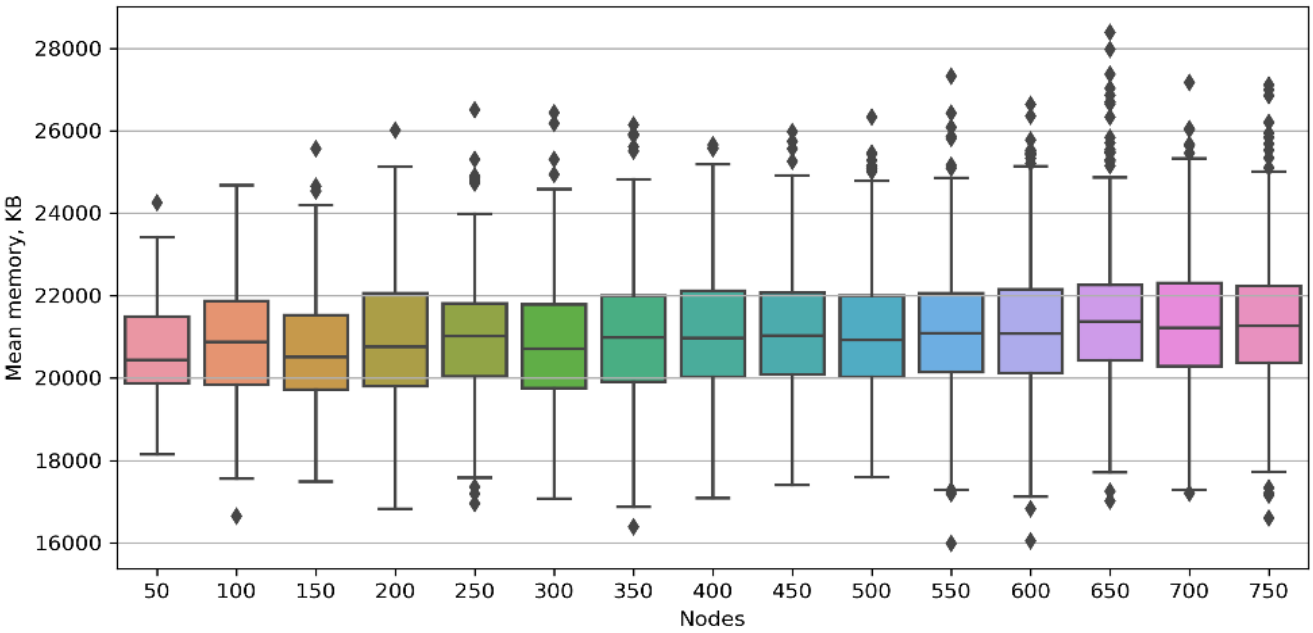


Рисунок 2.3 - «Box plot» розподіл середнього використання пам’яті вузлами у мережах різного розміру, топологія випадкового дерева

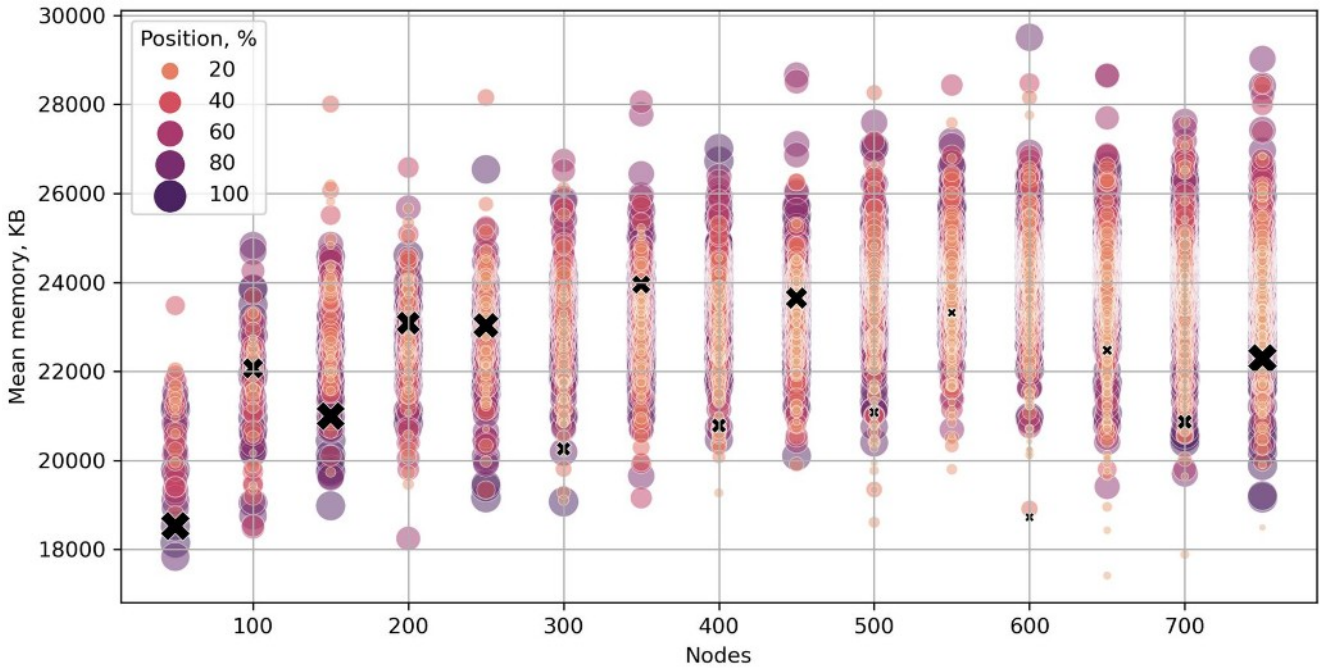


Рисунок 2.4 - Середнє використання пам’яті кожного вузла у мережах різного розміру, лінійна топологія

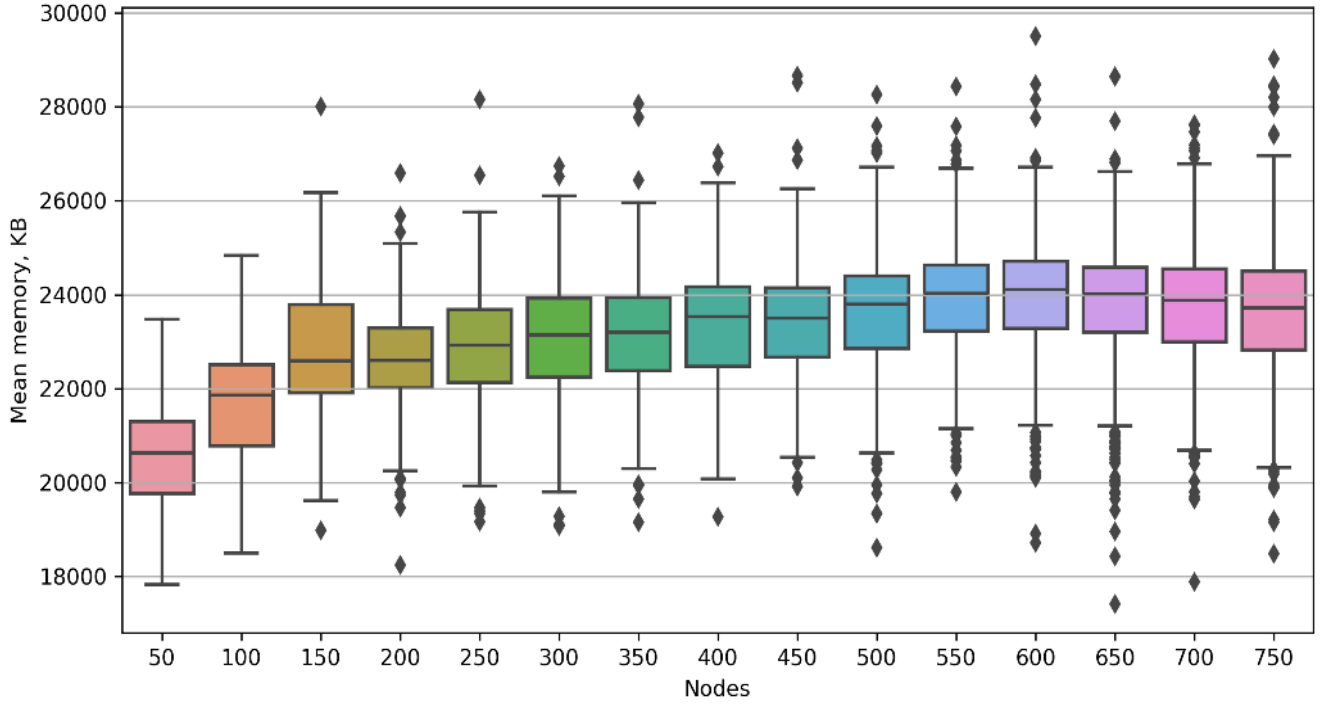


Рисунок 2.5 - «Box plot» розподіл середнього використання пам’яті вузлами у мережах різного розміру, лінійна топологія

Таблиця 2.1 - Пікове використання пам’яті

Розмір мережі	Пікове використання пам’яті, КБ		Розмір мережі	Пікове використання пам’яті, КБ		Розмір мережі	Пікове використання пам’яті, КБ	
	Дерево	Лінія		Дерево	Лінія		Дерево	Лінія
50	28640	26640	300	29300	30300	550	28664	30740
100	27980	27740	350	28812	31840	600	28624	32712
150	29036	29496	400	29004	30512	650	30092	32116
200	27432	28104	450	27736	30812	700	29620	30912
250	28660	29912	500	29428	31168	750	29564	30656

Іншу динаміку для лінійної топології можна спостерігати на графіку середнього використання процесору у порівнянні з положенням вузла, показаному на рисунку 2.6. Коли вузли розташовані в лінію, використання процесору масштабується відповідно до її довжини і тим більше, чим ближче до середини лінії конкретний вузол. Крім того, кореневий вузол завжди ділить лінію на дві частини, для яких довжина та середні точки розглядаються окремо. Це особливо чітко видно на лініях у 700 та 450 вузлів, де кореневий вузол опинився ближче до середини. У першому випадку значно менше вузлів ліворуч від кореня, ніж праворуч від нього, що призводить до значної різниці у використанні процесору, зберігаючи при цьому тенденцію більшого використання процесору середніми вузлами для обох сторін. Топологія з 450 вузлами має подібний випадок, за винятком того, що цього разу вона поділена на приблизно рівні частини, що дозволяє їй мати загально менше використання процесору, ніж в меншої мережі з 400 вузлів. Це спостереження приводить до висновку, що більш вигідно тримати корінь мережі ближче до її фактичного центру, тобто так, щоб кількість вузлів на кожній гілці була приблизно однаковою. Одним із способів цього можна досягти є цілеспрямований «майнінг» малих ключів.

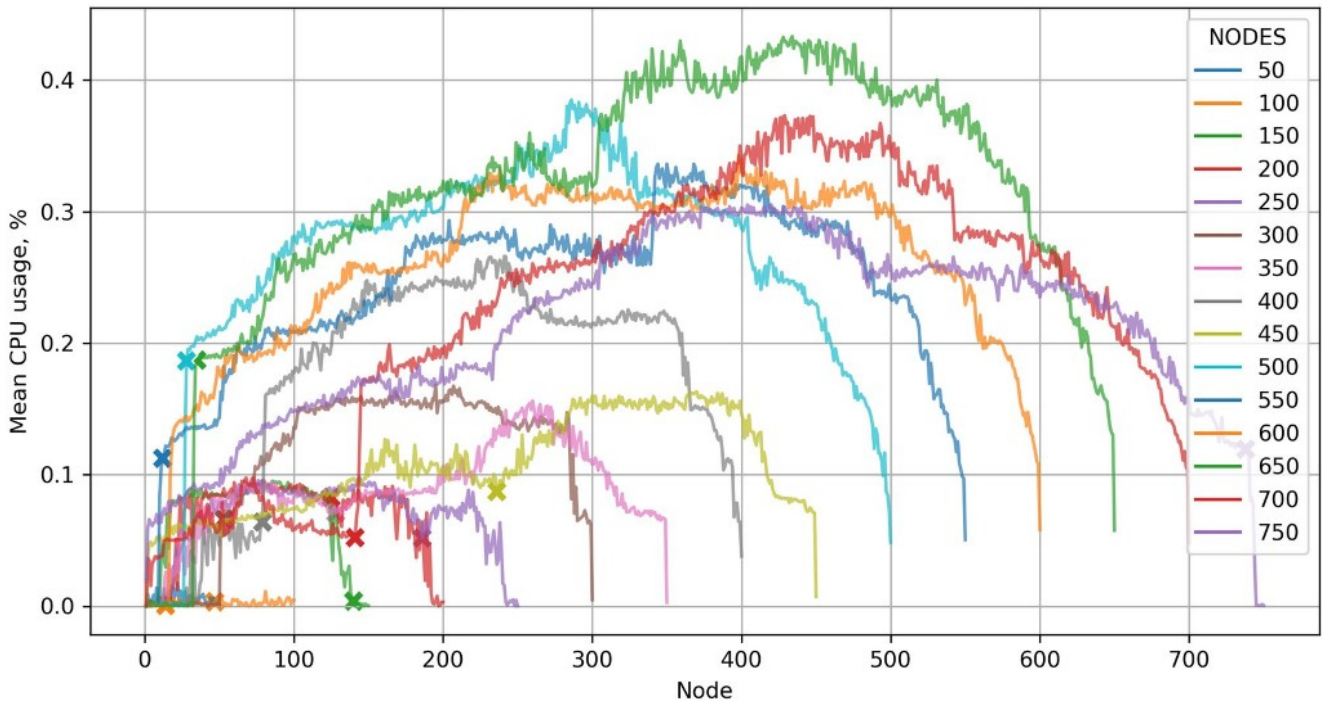


Рисунок 2.6 - Середнє використання процесору кожним вузлом у лінійній топології для мереж різних розмірів, хрестами відмічено кореневі вузли

Примітка: Використання процесору записується за допомогою стандартної утиліти Linux ps. Згідно з її сторінкою мануалу [20], ця метрика наразі виражається нею як відсоток процесорного часу від усієї тривалості життєвого циклу процесу. Це не ідеально, але підходить для цього випадку використання, оскільки дозволяє порівнювати процеси демона маршрутизації, які запускаються та завершуються одночасно.

На рисунку 2.7 представлено надходження пакетів за часом на лінійній топології для мереж різних розмірів. Точки лінії трохи зміщені на осі часу через те, що пінги мають довший час проходження на довших топологіях. Мережі з 700 і 750 вузлами ніколи не досягають 100% надходження пакетів. Після подальше дослідження виявило, що це відбувається через повернення утилітою ping «змішаного» результату середовищу meshnet-lab – у довгих топологіях потрібно чекати значно довше, поки прийде початковий пакет, оскільки спочатку виконується пошук вузла. Це призводить до того, що іноді вузол надсилає другий

пакет, який успішно надходить і призводить до того, що результат становить «50% прибуття».

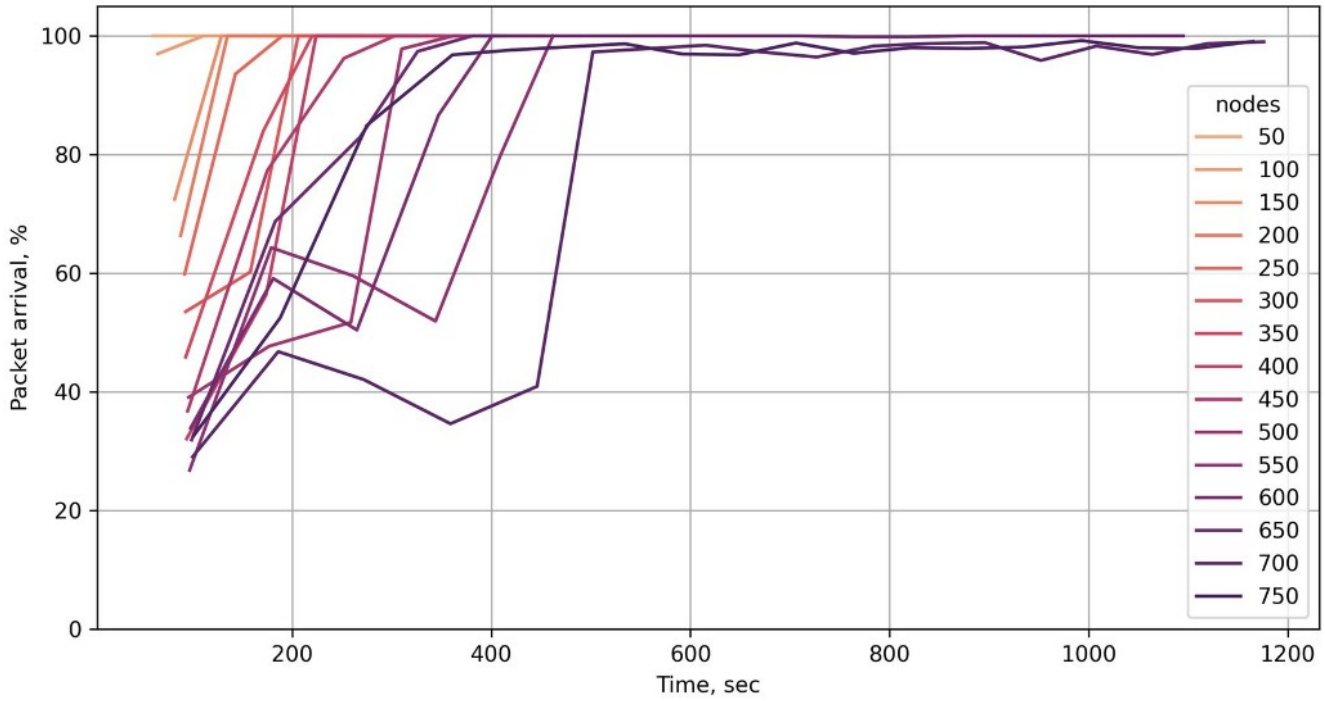


Рисунок 2.7 - Надходження пакетів для мереж різних розмірів, лінійна топологія

Маючи цю інформацію, ознакою конвергентної мережі можна вважати надходження пакетів вище 95%, та таким чином дізнатися приблизний час конвергенції, представлений у таблиці 2.2. Слід зазначити, що вимірювання були записані з інтервалом принаймні 60 секунд одне від одного, тому ці числа не слід приймати за номіналом. Однак їх можна використати для побудови графіка часу конвергенції, показаного на рисунку 2.8. Отриманий графік передбачає лінійну залежність між довжиною мережі та часом конвергенції.

Таблиця 2.2 - Час конвергенції на лінійній топології

Довжина мережі	Час конвергенції, с	Довжина мережі	Час конвергенції, с	Довжина мережі	Час конвергенції, с
50	60	300	206	550	462
100	65	350	220	600	401
150	129	400	224	650	326
200	135	450	252	700	502
250	190	500	310	750	361

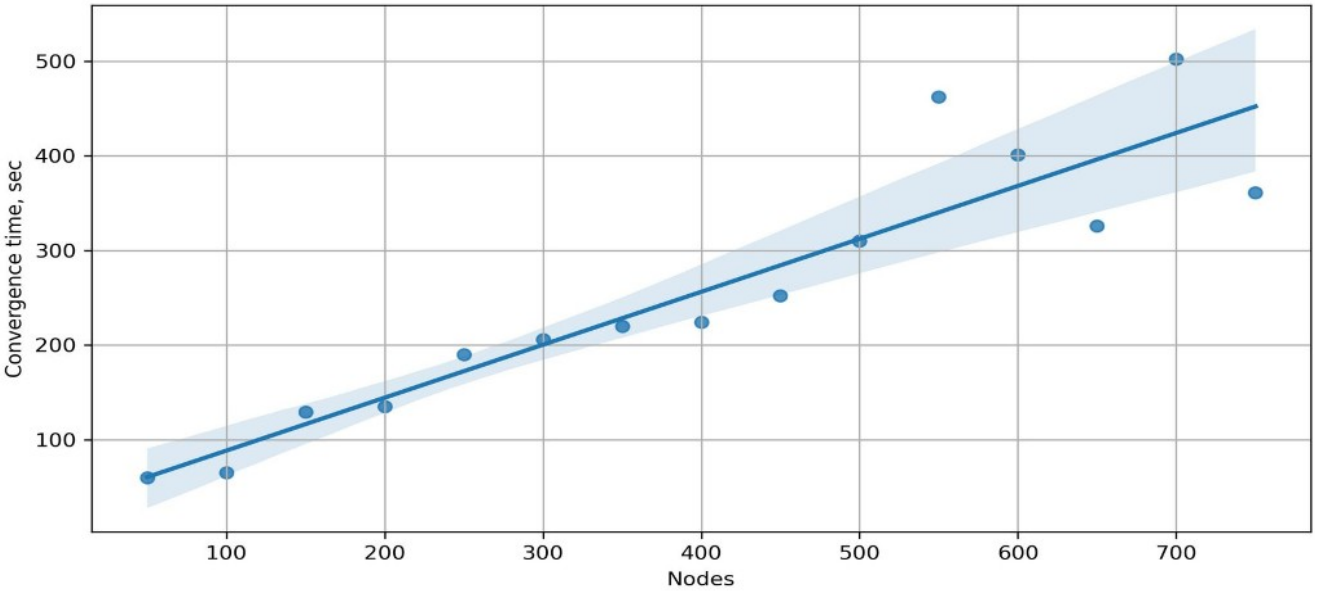


Рисунок 2.8 - Час конвергенції на лінійній топології різної довжини

Для порівняння, той самий експеримент на лінійній топології лінії проводився з різними протоколами маршрутизації (OLSR, Babel) і показав, що вони не здатні працювати з довгими мережами – надходження пакетів падає нижче 80% на 150 вузлах і нижче 40% на 450 (рис.2.9). Також проведені подовжені експерименти, які показали, що надходження пакетів для цих протоколів ніколи не збільшується.

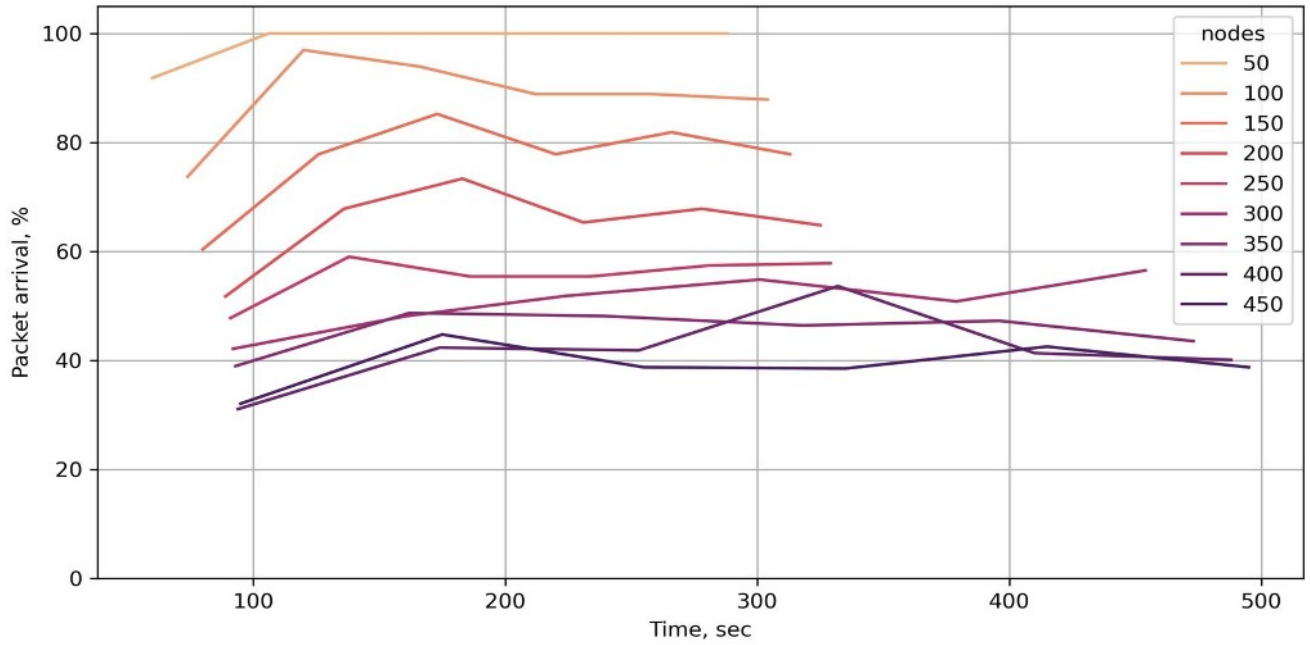


Рисунок 2.9 - Надходження пакетів для мереж різних розмірів, лінійна топологія, протокол OLSR

Використання пам’яті за часом в лінійній топології на 750 вузлів представлено на рисунку 2.10. Лінії мають зменшену прозорість, щоб показати загальну тенденцію, яка свідчить про константне використання пам’яті після досягнення мережею конвергенції.

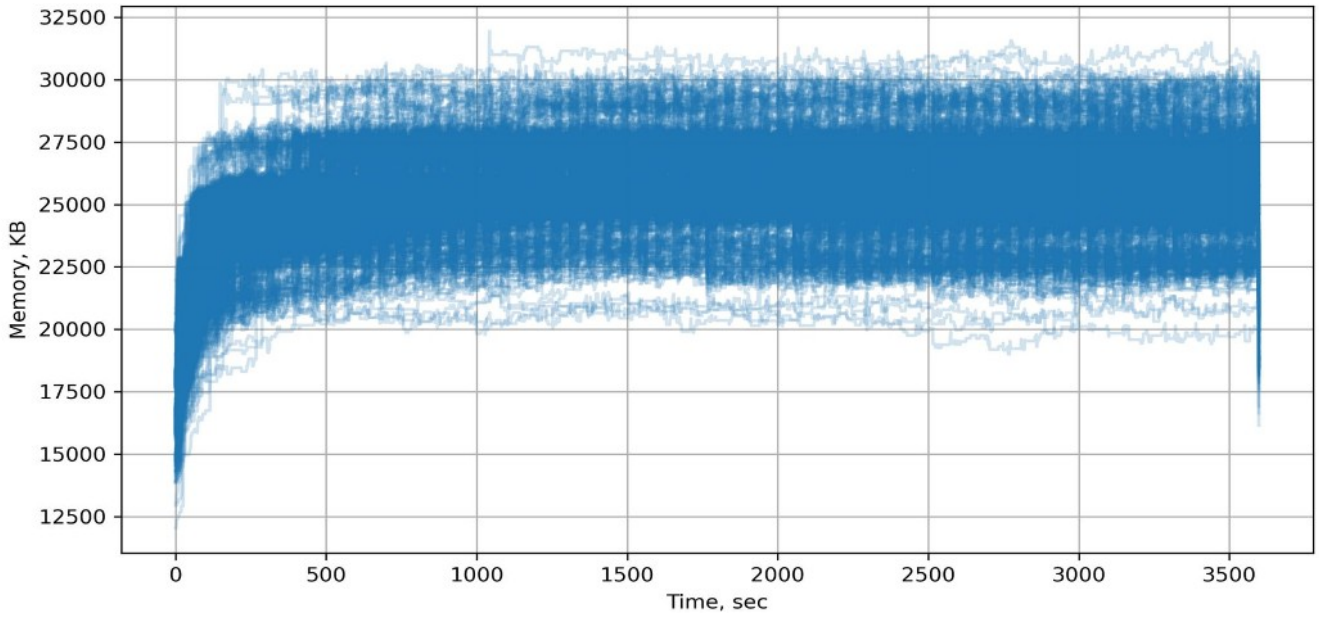


Рисунок 2.10 - Використання пам’яті за часом, лінійна топологія, 750 вузлів

Результати вимірювання використання ресурсів процесору на топології випадкового дерева з 50 вузлами без обмежень пропускної здатності наведено на рисунку 2.11. Графік демонструє, що вузли, які приймають та передають дані мають найбільше та друге найбільше значення використання процесору відповідно. За ними йдуть проміжні вузли, та решта вузлів, які не беруть участі в передачі, що підтверджує припущення, зроблене раніше. Значення вимірювань використання ресурсу процесора зростають під час передачі, а потім поступово зменшуються через специфіку застосування утиліти ps.

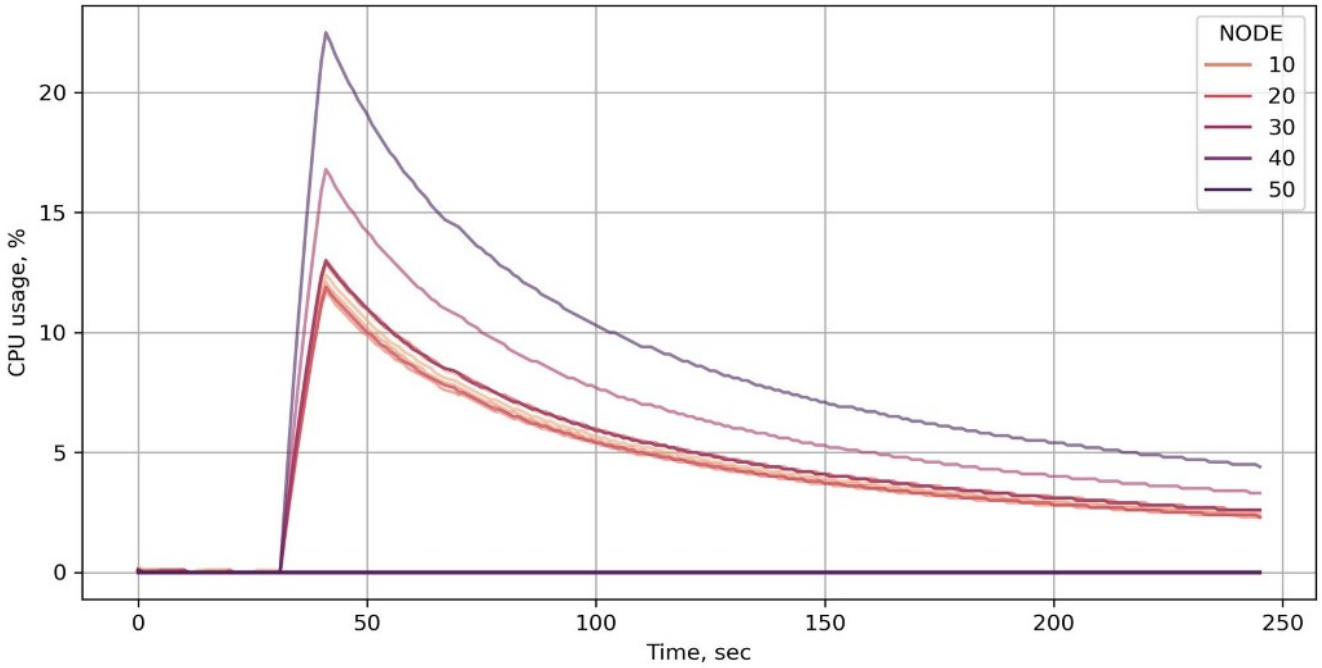


Рисунок 2.11 - Використання процесору за часом, топологія випадкового дерева, 50 вузлів, трафік від вузла 001e до 0032 (30 та 50)

Експеримент на лінійній топології на 750 вузлів з повільними з’єднаннями показав, що зв’язок із 749 переходами все ще можливий, хоча й зі значною затримкою від 18 до 20 секунд, що вимагає використання в цій мережі додатків, стійких до затримок.

2.2 Маломасштабне тестування

2.2.1 Методи

Метою маломасштабного тестування є отримання даних про продуктивність Yggdrasil у складніших мережах, ніж просто велика статична мережа. Через це процес тестування має більш дослідницький характер. Загалом, нам потрібно було побачити, як саме будується остовне дерево Yggdrasil, якими шляхами йде трафік і наскільки допустимою є мобільність вузлів. Ці дані є важливими, оскільки вони розкривають потенційні особливості застосування та недоліки схеми маршрутизації. Для цього автором збиралася деревоподібна топологія із вже відомим коренем. Після запуску демонів маршрутизації Yggdrasil до цієї топології вводилися додаткові з'єднання. Після отримання такої мережі проводився аналіз методом опитування демонів маршрутизації про їх представлення дерев, та запускались різні однобічні потоки трафіку, щоб побачити шляхи передачі даних.

Тести мобільності проводилися шляхом переключення одного блукаючого вузла між статичними вузлами, одночасно пінгуючи інший вузол. Особливо цікавими є випадки обрання вузлом нового батьківського вузла після втрати зв'язку із поточним та випадок мережі із блукаючим кореневим вузлом.

Випробування проводилися в середовищі coreemu-lab [18], оскільки воно є «batteries-included» дистрибутивом у вигляді Docker-образу Common Open Research Emulator (CORE) [19], що розширений функціями автоматичного тестування моніторингу вузлів. На відміну від meshnet-lab, це програмне середовище, для легкої конфігурації мережі та взаємодії з вузлами використовує графічний інтерфейс користувача, а також декілька типів вузлів і з'єднань, включаючи кабельні та бездротові з'єднання в локальну мережу. Остання функція є особливо корисною, завдяки можливості динамічно формувати та розривати з'єднання методом простого переміщення вузлів під час емуляції мережі. Цей процес переміщення вузлів також можна автоматизувати за допомогою скриптів

мобільності. Крім того, іншою корисною функцією CORE є візуалізація трафіку дротових з'єднань і бездротових інтерфейсів, яка дозволяє відстежувати його потоки.

Наданий образ Docker модифіковано, щоб включити демон маршрутизації Yggdrasil. Наступні тести виконувались із попередньо згенерованими ключами для кожного вузла з метою кращого контролю над остовним деревом Yggdrasil.

2.2.2 Результати

Перший маломасштабний експеримент проведено на лінійній топології із 9 вузлами, де вузол n1 встановлював зв'язки із декількома вузлами цієї лінії. Мережа та відповідна послідовність дій наведені на рисунку 2.12. Фіолетові стрілки вказують на структуру остовного дерева Yggdrasil у формі зв'язків «нащадок вузлу», де n9 є кореневим вузлом.

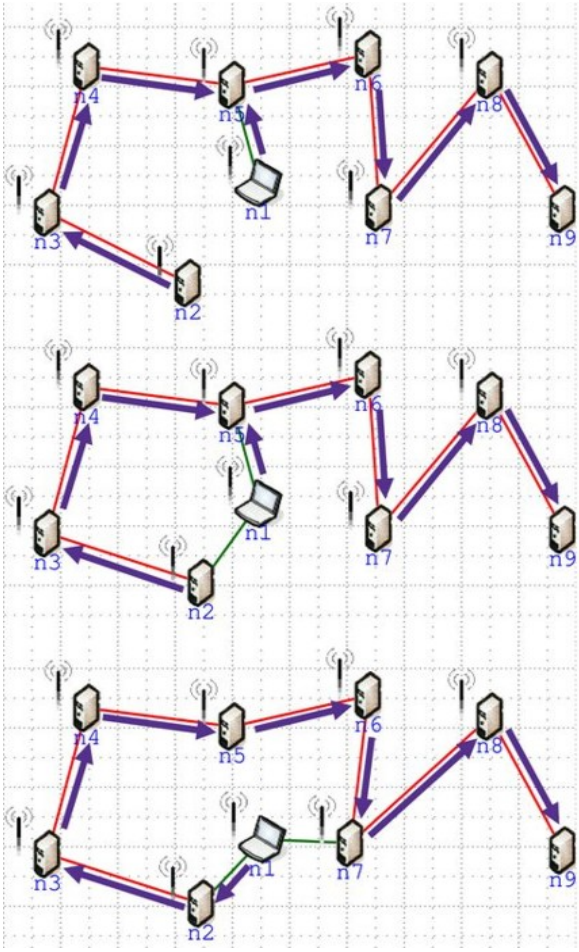


Рисунок 2.12 - Лінійна мережа на 9 вузлів, вузол n1 встановлює різні з'єднання

Спочатку, n1 з’єднаний лише з n5, який є його дочірнім вузлом. Потім з’єднання з n2 і n7 встановлюються саме в такому порядку. Коли з’єднання n1 – n5 розривається, n1 постійно вибирає n2 у якості нового батьківського вузла як з’єднання з найдовшим часом безвідмовної роботи, незважаючи на те, що n7 знаходиться ближче до кореня. Хоча це може здаватися неефективним, така логіка підтримує стабільність мережі, залишаючи в остовному дереві лише найстабільніші зв’язки, роблячи його таким чином «хребтом» мережі. Також стає зрозуміло, як саме будується остовне дерево – ключі мають значення лише при виборі кореневого вузла, що стає точкою відліку для решти мережі. Остовне дерево збирається навколо кореня на основі зв’язків, які встановлюються першими та розриваються останніми.

Наступний експеримент проводився на бездротовій mesh мережі розміром у 19 вузлів. Вузол n1 посилав пінг на віддалений вузол n20 великими пакетами, через що середовище визначало ці вузли, як вузли із значним об’ємом трафіку (рис. 2.13).

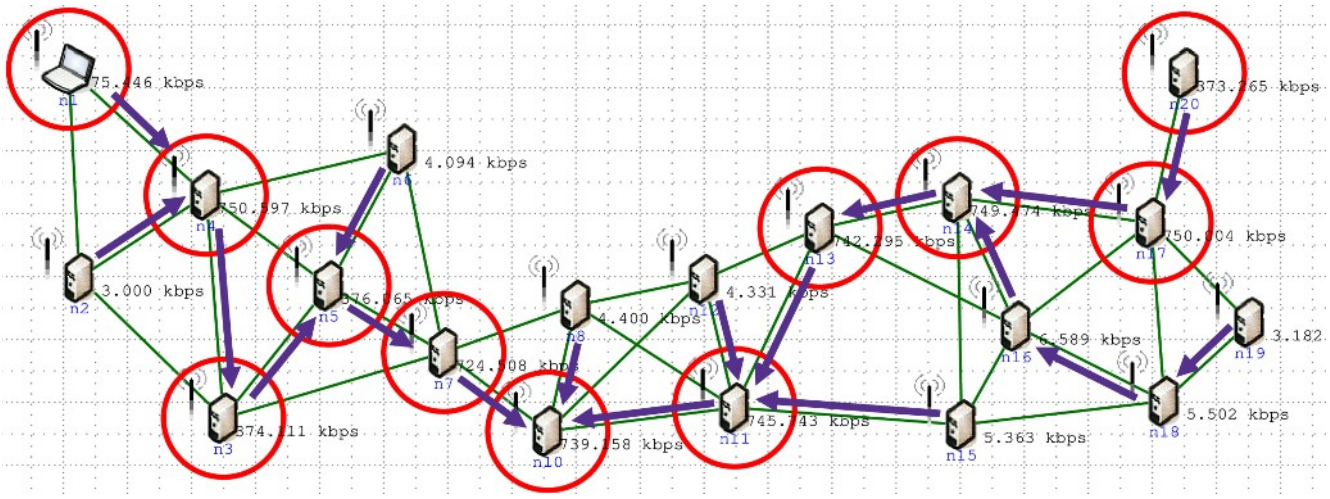


Рисунок 2.13 - Бездротова mesh мережа на 19 вузлів, трафік між n1 та n20

Звертаючи увагу на обсяги трафіку, які вказані біля кожного вузла, можна помітити, як після вузла n2 трафік нібито розподіляється між вузлами n3 і n5 знижуючи швидкість (з 750 Кбіт/с до 375 Кбіт/с). Подальше дослідження надало змогу визначити, що це є наслідком жадібної маршрутизації, через яку потоки

трафіку n1 → n20 і n20 → n1 передаються різними шляхами. Коли трафік передається від n1 до n20, використовується короткий шлях n4 – n5, оскільки він дозволяє пропустити вузол n3. Таким же чином для трафіку, що передається від вузла n20 до n1, використовується з'єднання між вузлами n7 – n3, оскільки воно дозволяє пропустити вузол n5.

Ця поведінка більш чітко простежується в топології, яка наведена на рисунку 2.14. В цій топології, для більшості вузлів використовуються кабельні з'єднання із односпрямованими потоками трафіку між вузлами n1 і n2. З'єднання поза деревом між вузлами n12 – n14 та n13 – n15 дозволяють трафіку пропускати два вузли і тому вважаються більш ефективними. Ця особливість використання різних шляхів для потоків трафіку між двома вузлами є корисною для збільшення пропускної здатності та зменшення заторів при передачі.

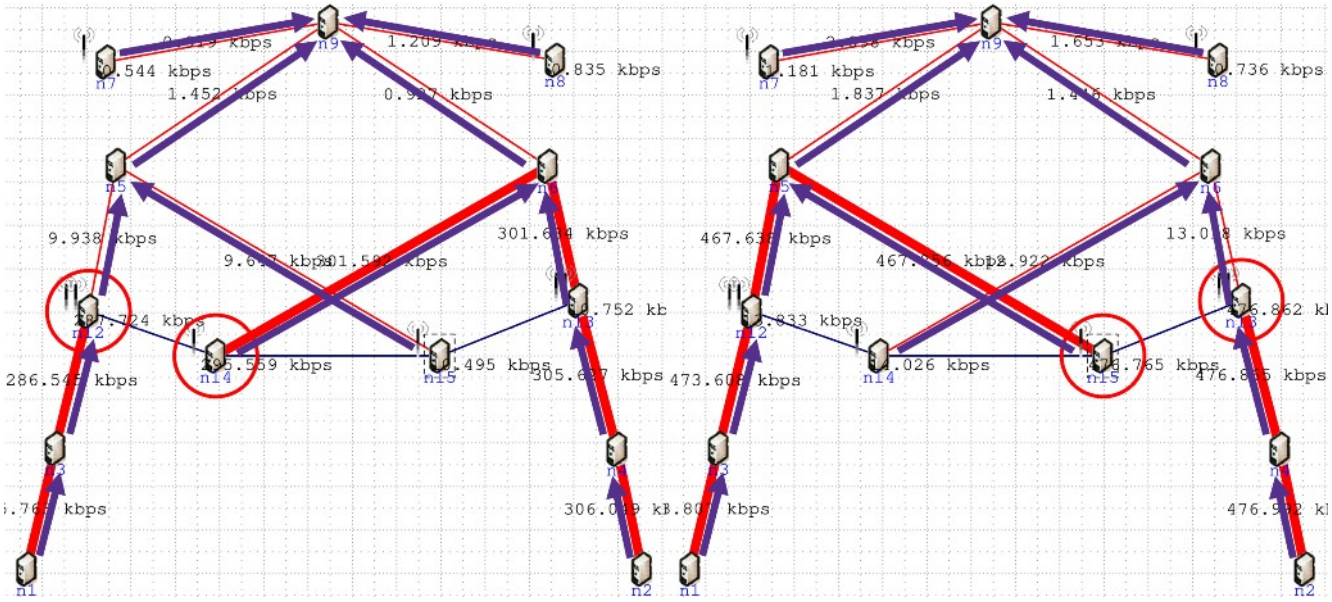


Рисунок 2.14 - Мережа на 15 вузлів, потоки трафіку n1 → n2 та n2 → n1

Однак жадібна маршрутизація має свої недоліки, головним чином у тому, що враховуються лише шляхи довжиною в один перехід (рис. 2.15). Зауважте, що мережа така ж, як і в попередньому прикладі, за винятком того, що вузли n14 і n15 переставлені місцями. Як наслідок, з'єднання вузлів n12 – n14 і n13 – n15 більше не

забезпечують жодних миттєвих переваг для вузлів n12 і n13, тому вони не використовуються, хоча взагалі зрозуміло, що шлях через n12, n15, n14 та n13 на один перехід коротше. Це призводить до того, що два потоки трафіку проходять однаково довший шлях через кореневий вузол.

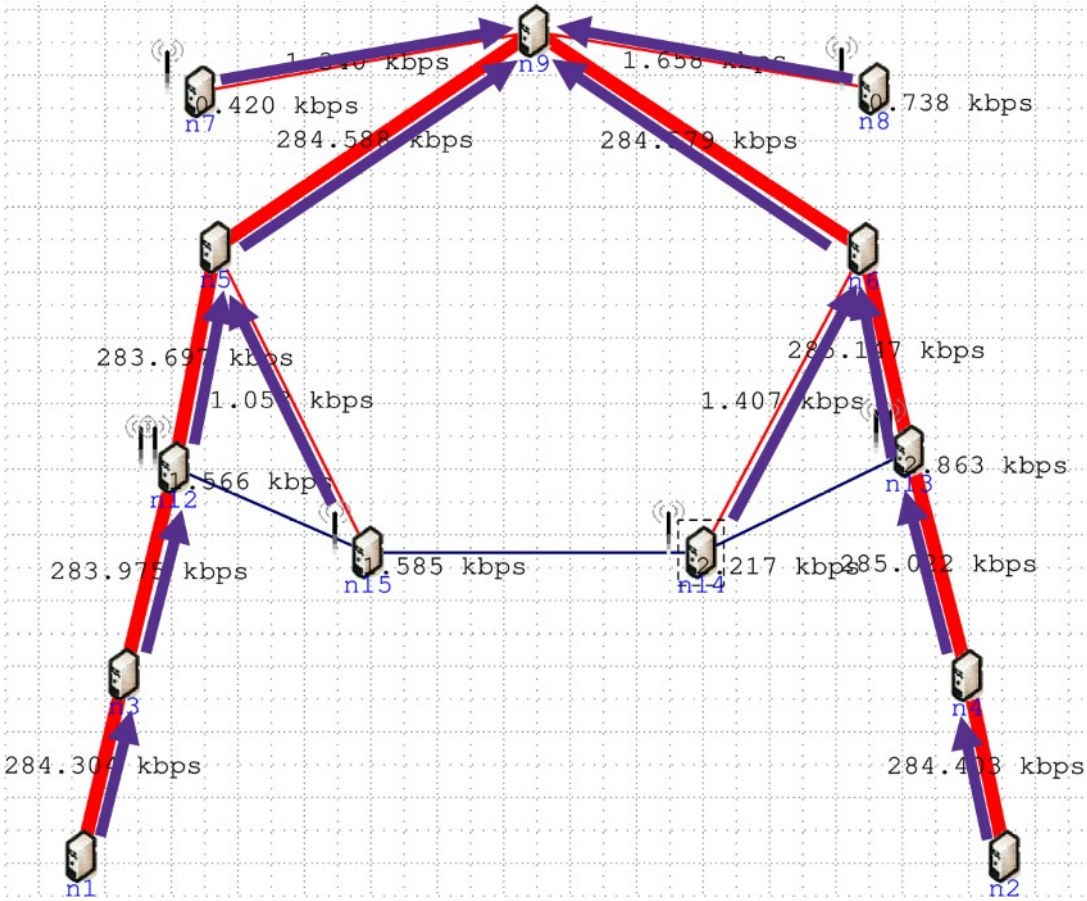


Рисунок 2.15 - Модифікована мережа на 15 вузлів, потоки трафіку n1 → n2 та n2 → n1

Автором проведено два експерименти з дослідження мобільності вузлів: з блукаючим «листовим» вузлом та блукаючим коренем у більшій мережі. У першому вузол n1 постійно переключається між листовими вузлами мережі, яка наведена на рисунку 2.16. Така поведінка не впливає на зв’язок у мережі, окрім зв’язку із самим вузлом n1. Дійсно, постійна зміна сусідів означає постійну зміну координат дерева, таким чином запобігаючи будь-яким тривалим потокам трафіку до блукаючого вузла або порушуючи їх через його пошук. Трафік, що надходить із

вузла n1, більш успішно досягає пункту призначення, враховуючи, що пункт призначення є статичним вузлом, оскільки його координати не змінюються після знаходження та протягом передачі. Щоб успішно отримувати трафік, блукаючий вузол має підтримувати принаймні одного постійного сусіда протягом передачі, інакше передача переривається процесом пошуку. Таким чином, Yggdrasil допускає мобільність вузлів, але не заохочує її та вимагає від вузлів підтримувати свої координати для двонаправленого обміну даними.

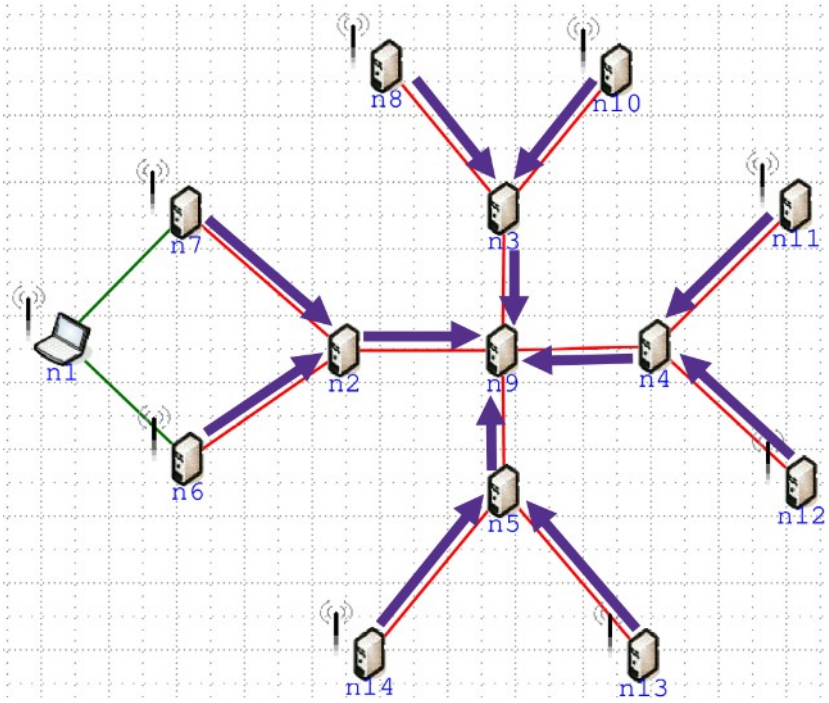


Рисунок 2.16 - Мережа на 14 вузлів, блукаючий вузол n1

Однак блукаючий кореневий вузол – це зовсім інший випадок. На рисунку 2.17 показано мережу, подібну до попереднього прикладу, за винятком того, що цього разу кореневий вузол n9 постійно переключається між парами листових вузлів із періодом 10 секунд. Це змушує вузли, до яких він підключається, змінювати свої координати у дереві та поширювати дані про цю зміну по всій мережі. Однак, до того часу, кореневий вузол може створювати з’єднання із іншими сусідніми вузлами, що також спричиняє поширення конфліктного перевпорядкування. Це призводить до того, що вузли мають різні представлення

про зв’язки у остовному дереві, що у свою чергу є причиною різних проблем з передачею трафіку (рис. 2.17). У такій мережі невеликого масштабу трафік між вузлами n18 і n26, протягом більшої частини часу передачі, обирає правильний маршрут, лише іноді порушуючи його. Але розумно очікувати серйозних збоїв у великій мережі, особливо коли нові кандидати на роль кореневого вузла з’являються в абсолютно різних частинах мережі. Це можна вважати потенційним напрямом атаки на мережу. У цьому випадку можливим способом зниження такого впливу на працездатність є майнінг малих ключів, що дозволяє налаштувати постійний кореневий вузол із попередньо згенерованим достатньо малим ключем, аж до того випадку, коли майнінг менших ключів стає просто невигідним. Також, із міркувань надмірності, є доцільним створювати декілька потенційних коренів.

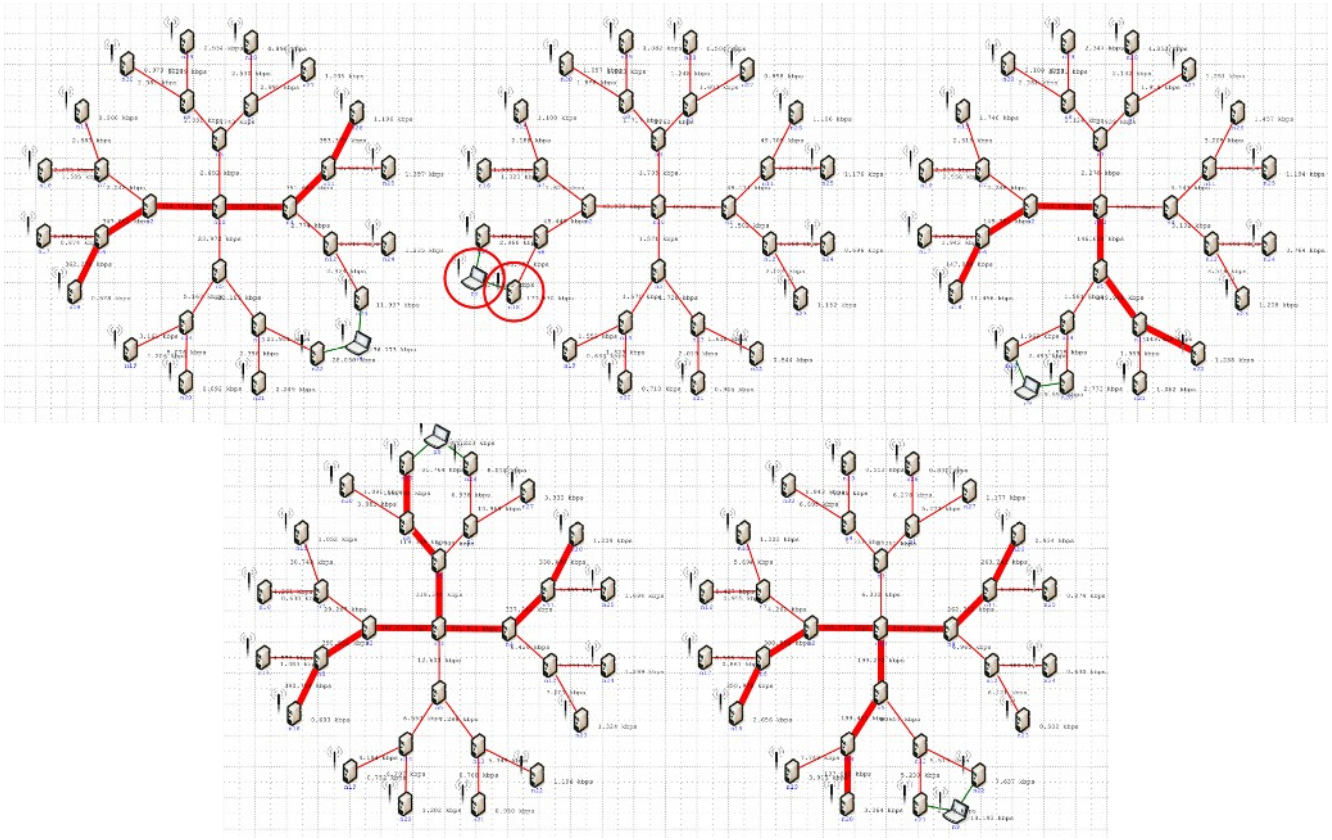


Рисунок 2.17 - Мережа на 30 вузлів, блукаючий кореневий вузол n9, нормальний трафік від n18 до n26 та його аномалії

2.3 Висновки

Було розглянуто питання масштабованості схеми маршрутизації Yggdrasil, для чого проведено низку експериментів як на великих, так і на малих топологіях із використанням середовищ meshnet-lab і coreemu-lab. Результати експериментів показали, що Yggdrasil значно перевершує у ліміті переходів інші широко використовувані протоколи mesh маршрутизації, такі як OLSR і Babel. Експерименти також показали типовий час конвергенції, а також тенденції використання пам’яті та процесора Yggdrasil і підтвердили їх масштабування з глибиною мережі, а не її загальним розміром. Це підтверджує доцільність використання Yggdrasil у якості основи для розгортання великомасштабних децентралізованих mesh мереж незалежних вузлів як альтернативи традиційним централізованим ієрархічним мережам, що використовуються в мережах інтернет-провайдерів.

Тестування також виявило деякі особливості у мережах на основі Yggdrasil, такі як збільшення використання процесору на довших гілках остовного дерева та аномалії трафіку, внесені блукаючим кореневим вузлом. Як засіб пом’якшення цих проблем було запропоновано упереджувальний майнінг малих ключів для детермінованого розміщення кореневих вузлів.

3 РОЗГОРТАННЯ МЕРЕЖІ

Маючи підтвердження можливості розгортання масштабної децентралізованої мережі на основі Yggdrasil, можна приступити до проєктування самої мережі. Для цього потрібно визначити:

- бажану базову технологію канального рівня;

- структуру мережі;
- операційну систему прототипу вузла мережі;
- необхідне для побудови вузла апаратне забезпечення.

3.1 Технології канального рівня

Технологія канального рівня має дозволяти сусіднім вузлам автономне встановлення сеансів передачі даних між собою, не спираючись на проміжні пристрої (точки доступу, комутатори, тощо). При цьому на мережевому рівні повинен підтримуватись протокол IPv6 і важливими є параметри пропускної здатності і розповсюдження технології. Для кабельних з'єднань ці вимоги вже підтримуються стандартами IEEE 802.3 на технологію із загальною назвою Ethernet. У випадку бездротового зв'язку можна розглянути декілька варіантів.

3.1.1 Стандарт Wi-Fi Direct

Wi-Fi Direct дозволяє двом і більше пристроям взаємодіяти без використання точки доступу, зазвичай використовується для обміну файлами з пристроями поблизу, трансляції зображення екрану на зовнішній монітор, бездротовий друк тощо.

Стандарт підтримується фактично усіма сучасними Wi-Fi пристроями і є надбудовою над звичайним Wi-Fi, тому на фізичному рівні працює абсолютно ідентично, підтримуючи ті ж самі частоти, схеми кодування, пропускну здатність, тощо. Також можлива робота у режимі сумісності із пристроями, які не підтримують Wi-Fi Direct. Вони можуть під'єднуватись до мережі, як до звичайної точки доступу.

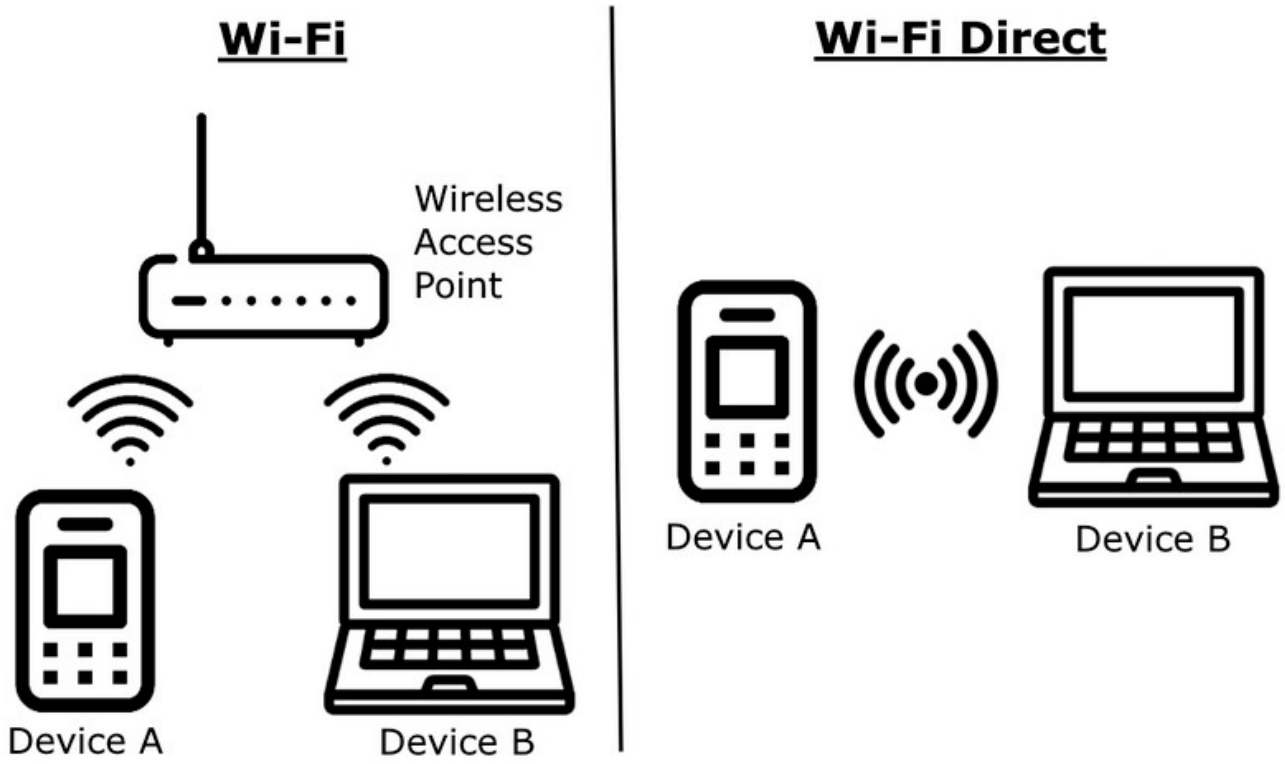


Рисунок 3.1 - Відмінність Wi-Fi Direct від інфраструктурного режиму Wi-FI

Недоліки стандарту виявлені в процесі проведення аналізу, основний недолік пов’язаний із реальним принципом роботи цього стандарту. Пристрої, що використовують Wi-Fi Direct, приймають на себе ролі «власника групи» (режим P2P-GO) та «клієнта» (режим P2P-client). При цьому мережу підтримує лише власник групи, виконуючи функцію точки доступу і комутатора, а всі інші пристрої є просто клієнтами. Фактично Wi-Fi Direct працює як звичайний режим інфраструктури з одним із пристроїв, який виконує роль точки доступу, тому не підходить у якості такої технології для mesh мережі. Деякі науковці [21] пропонували поєднувати групи, використовуючи власника групи одночасно як клієнта іншої мережі, але це рішення є менш практичним (рис 3.2).

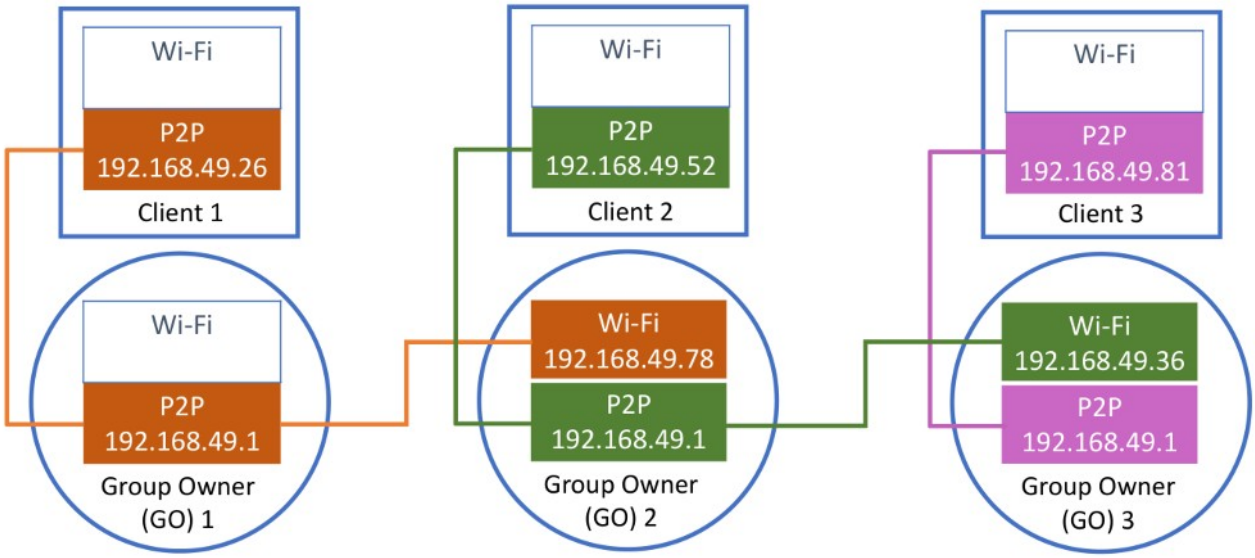


Рисунок 3.2. - Поєднання Wi-Fi Direct груп через власників [21]

3.1.2 Wi-Fi ad-hoc (режим IBSS)

Wi-Fi також підтримує режим «ad-hoc» або IBSS (Independent Basic Service Set), який, на відміну від Wi-Fi Direct, реалізовує справжню однорангову мережу. Вузли можуть обмінюватися даними напряму, знаходячись у радіусі дії один одного. Але такий варіант має недолік у вигляді жорсткого обмеження на пропускну здатність у 11 Мбіт/с.

3.1.3 Режим 802.11s

Подібно до Wi-Fi Direct, режим 802.11s або Mesh Point є надбудовою над звичайним інфраструктурним режимом роботи Wi-Fi, що дозволяє вузлам об’єднуватися у mesh мережу і обмінюватися даними як з вузлами у зоні досяжності, так і поза нею. Протоколами мережевого рівня така mesh мережа сприймається як великий комутатор, що з’єднує вузли між собою.

Режим перш за все призначений для бездротового поєднання у локальну мережу кількох точок доступу, кожна з яких може також підтримувати свою мережу в режимі інфраструктури, а деякі є mesh «порталами», забезпечуючи вихід у зовнішню мережу/Інтернет (рис. 3.3).

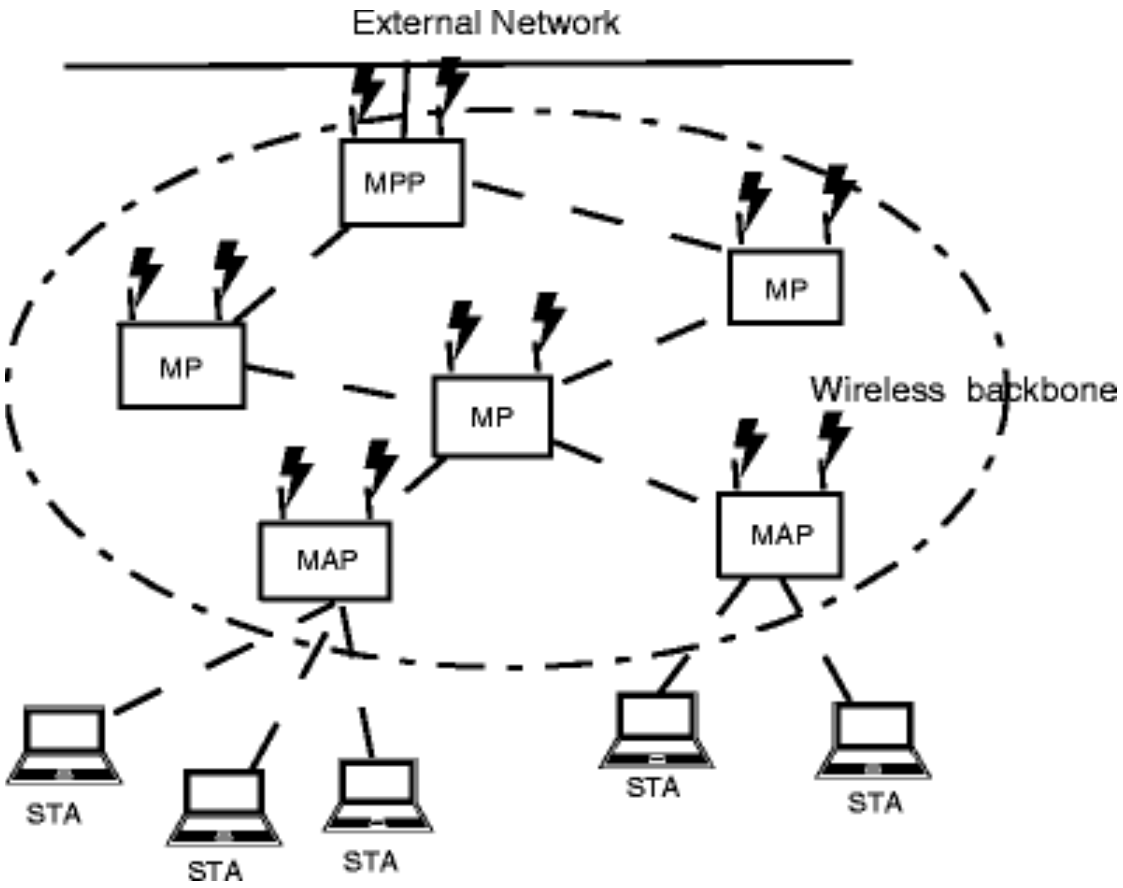


Рисунок 3.3 - Типове застосування та конфігурація мережі з 802.11s

За замовчанням використовується протокол маршрутизації HWMP (Hybrid Wireless Mesh Protocol). Перевагою 802.11s є можливість відключити застосування HWMP з метою використання інших протоколів маршрутизації, в цьому випадку вузли спілкуються між собою лише у зоні доступності один одного. При цьому доступний весь інший функціонал Wi-Fi: повний діапазон частот, розширення смуги пропускання, тощо. Основним недоліком є необхідність роботи всієї мережі на одній частоті. Хоча, якщо пристрій має декілька окремих бездротових інтерфейсів та антен, ніщо не заважає йому стати учасником mesh мережі на декількох частотах одночасно.

Виходячи з цього, очевидно, що стандарт Wi-Fi IEEE 802.11s із вимкненим стандартним протоколом маршрутизації є найкращим для використання у якості технології канального рівня.

3.2 Структура мережі

Технологія канального рівня частково визначає загальну структуру чи архітектуру мережі. Справа у тому, що більшість мобільних пристроїв є доволі закритими та або не підтримують стандарт 802.11s, або просто не дозволяють використовувати цей режим. Додатковою проблемою є наявність недостатнього покриття, якщо мережа застосовує лише клієнтські пристрої. В такому випадку потрібно будувати мережу на спеціальних вузлах, які можна переконфігурувати відповідно до умов використання, наприклад: у режим точки доступу; мосту зі спрямованою антеною; кабельного Ethernet шлюзу, тощо.

Таке рішення призведе до того, що в мережі будуть присутні вузли, які здатні одночасно встановлювати лише одне з’єднання. Тому таке рішення є неоптимальним, бо не вирішує завдання по розширенню покриття мережі. На жаль, на даний час не існує іншого способу під’єднати до мережі пристрої під управлінням операційних систем Windows, Android та IOS, хоча деякі клієнтські пристрої все ж таки під’єднуються до мережі за допомогою Wi-Fi адаптерів із підтримкою 802.11s, і таке можливо під управлінням систем Linux, BSD тощо. Альтернативне підключення також можливе із використанням спеціального вузла у якості шлюзу для пристрою. Таким чином, клієнтські пристрої допомагатимуть мережі створювати належне покриття, а не лише використовуватимуть її для передачі даних.

3.3 Операційна система

Демон Yggdrasil написано мовою програмування Go, тому він є архітектуро- і платформонезалежним, але все одно використовується у складі операційної системи, яка забезпечує мережевий стек протоколів, виділення пам’яті, взаємодію з користувачем тощо. До того ж, для роботи вузла необхідним є застосування

додаткового функціоналу: фаєрвол (мережевий екран), маршрутизація, DHCP сервер для режиму точки доступу, програмне забезпечення для віддаленого доступу тощо. Для вбудованого мережевого обладнання такий функціонал цілком забезпечує операційна система OpenWRT (Open Wireless Router) на основі ядра Linux.

Основними перевагами OpenWRT є її універсальність, модульність та широкі можливості з модифікації. Проєкт підтримує безліч різноманітних бездротових маршрутизаторів, модулів, плат розробки (development board), надаючи інструменти та профілі формування прошивок. Зручності додає система керування пакетами програмного забезпечення, які можна як включати у прошивку, так і встановлювати у процесі використання пристрою. Демон Yggdrasil, його утиліта моніторингу та керування yggdrasilctl є доступними у вигляді подібного пакету із офіційного репозитарію OpenWRT.

OpenWRT дозволяє опціонально включити у прошивку web-інтерфейс LuCI для налагодження і спрощеної взаємодії із користувачем. А також, окремим пакетом до нього можна включити функціонал конфігурації демона Yggdrasil.

Система формування прошивки Image Builder на основі утиліти make також дозволяє включати у прошивку окремі додаткові файли. Ця можливість використовується для додавання скриптів uci-defaults, які виконуються лише при першому завантаженні пристрою після прошивки або скиданні до заводських налаштувань, після чого вони видаляються.

Таким чином маємо можливість часткової автоматизації розгортання образу OpenWRT на пристрої. Web-інтерфейс, демон Yggdrasil;, засоби його конфігурації та інші пакети, разом зі скриптом автоматичного налаштування додаються в прошивку на етапі її формування. Після прошивки пристрою за допомогою протоколу TFTP і першого завантаження він одразу стає активним і готовим до роботи.

3.4 Апаратне забезпечення

До апаратного забезпечення маємо наступні технічні вимоги, це є:

- підтримка OpenWRT;

- наявність принаймні одного Wi-Fi інтерфейсу з підтримкою 802.11s;

- наявність хоча б 128 МБ оперативної та 16 МБ Flash пам’яті (виходячи з використання пам’яті системою та Yggdrasil);

- наявність Ethernet інтерфейсу;

- наявність інтерфейсу UART для початкового налагодження.

Цим вимогам задовольняє наведений модуль HLK-7628N виробництва Hi-Link (рис. 3.4). Модуль є фактично готовим бездротовим маршрутизатором на процесорі (системі на кристалі, SoC) MT7628N архітектури RAMIPS виробництва Mediatek. Його характеристики і параметри: 128 МБ оперативної та 32 МБ Flash пам’яті, Wi-Fi інтерфейс 2.4 ГГц на дві антени, п’ять Ethernet інтерфейсів, два інтерфейси UART, інтерфейси SPI, I2C, I2S тощо.

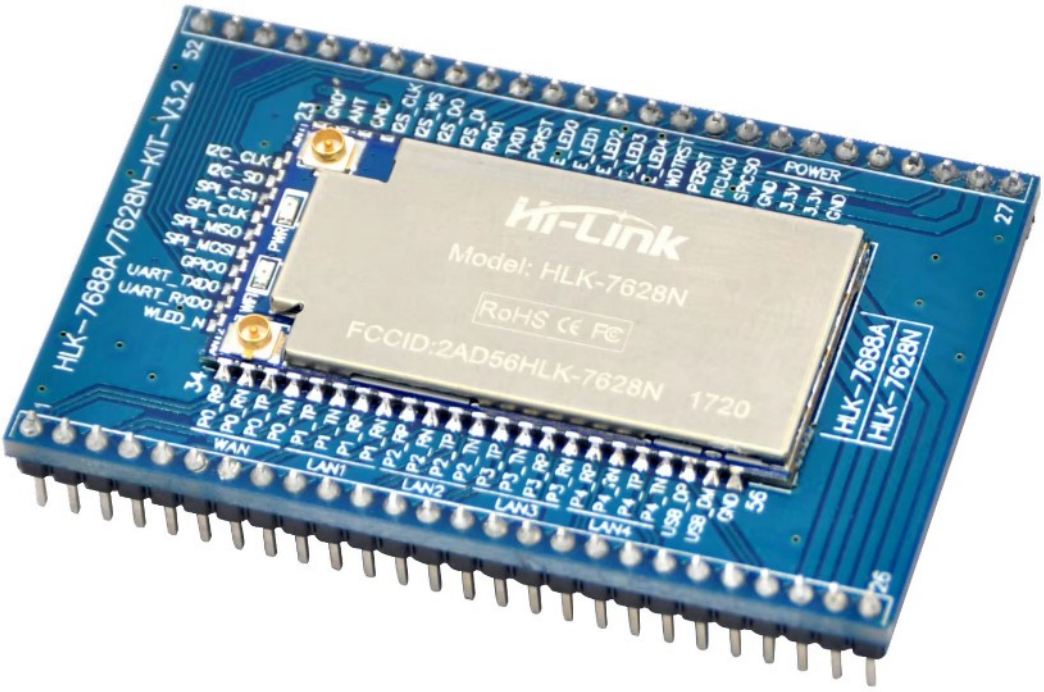


Рисунок 3.4 - Модуль HLK-7628N

Модуль потребує живлення напругою 3.3 В зі споживанням у 800 мА. Для роботи достатньо під’єднати антени до роз’ємів U.FL, з’єднати із відповідними виводами Ethernet роз’єм, кнопку Reset та USB-UART адаптер і подати живлення.

HLK-7628N поставляється у двох варіантах — лише сам модуль та з перехідною платою (рис 3.4). При виконанні пристрою на печатній платі використовувати перехідну плату необов’язково.

Для прототипу вузла знадобляться 2 штирові SMA антени на 6 dBi (рис. 3.5), перехідні кабелі SMA-U.FL (рис. 3.6), Ethernet роз’єм 8P8C з вбудованим трансформатором (рис. 3.7), 2 конденсатори на 10 нФ, USB-C роз’єм для живлення (рис. 3.8) та DC-DC перетворювач з 5 В на 3.3 В (рис. 3.9). Так пристрій можна підключати до блоку живлення, зарядного пристрою або павербанку за допомогою звичайного USB-C кабелю.

Таким чином, для прошивки та налагодження, додатково потрібен USB-UART адаптер (рис. 3.10) і, оскільки прототип збирається на макетній платі, ще знадобляться дроти-перемички (джампери).

Рисунок 3.5 - Штирові SMA антени на 6 dBi

Штирові антени найбільше підходять для потенційно мобільних вузлів, оскільки мають всеспрямовану діаграму спрямованості і, як наслідок, найменшу вірогідність втратити зв’язок з сусіднім вузлом при переміщенні. Якщо ж вузол статичний і заздалегідь відомо, де і для яких з’єднань він встановлюється, можливо встановлення спрямованих антен замість всеспрямованих штирових. Вони дозволять значно підвищити дальність і якість зв’язку у певному напрямку за рахунок інших. Це має сенс при встановленні з’єднань між частинами мережі там, де штирових антен не вистачає, а прокладати кабель незручно або і зовсім неможливо, наприклад, при зв’язку через широку автомобільну магістраль або річку. Використання спрямованих антен також зменшує вплив завантаженості використовуваного частотного діапазону на якість з’єднання шляхом зменшення вірогідності колізій.

Можливо також і поєднання різних типів антен навіть з одним бездротовим інтерфейсом. Наприклад, на одному пристрої одночасно можуть використовуватись штирова та спрямована антена для з’єднань з вузлами поблизу та на значній відстані у певному напрямку відповідно.

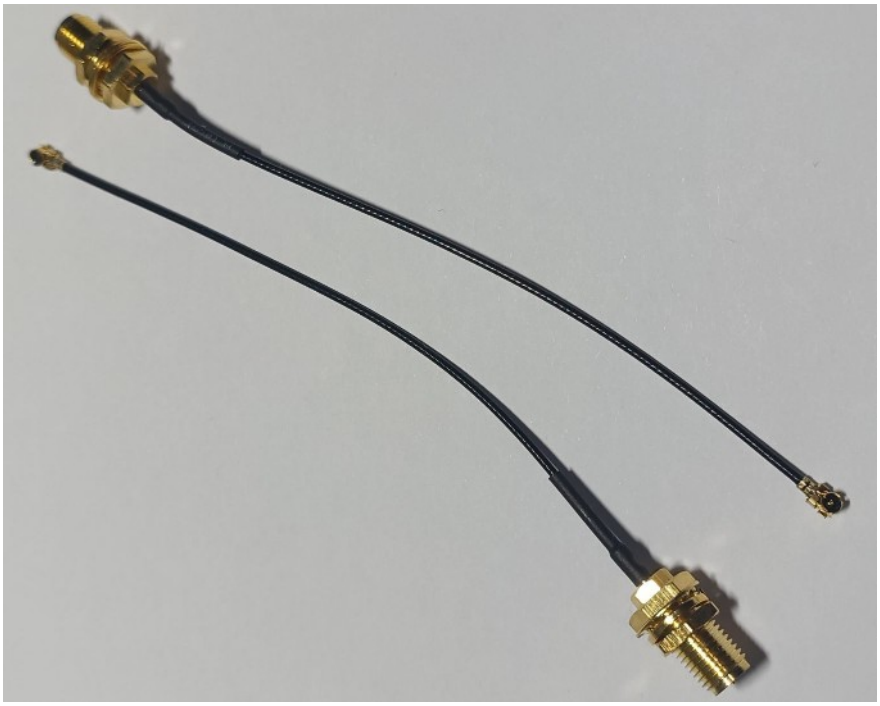


Рисунок 3.6 - Перехідні кабелі SMA-U.FL

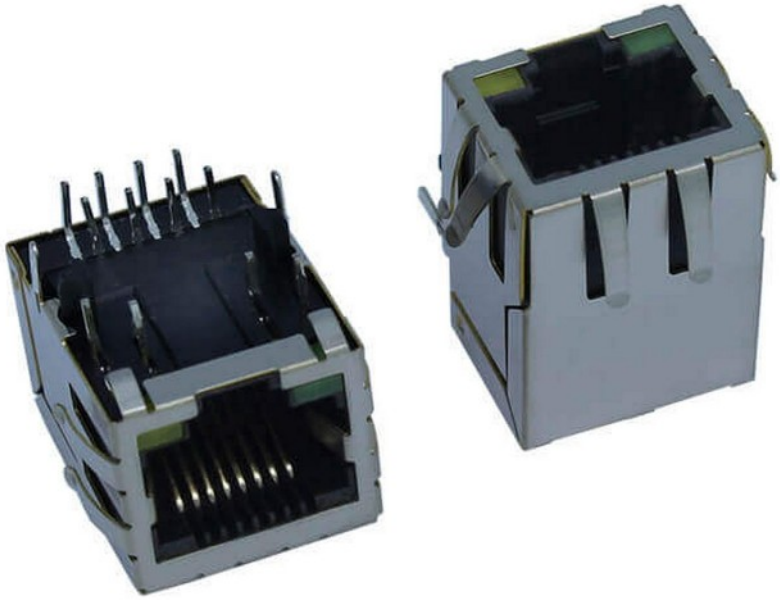


Рисунок 3.7 - Ethernet роз’єм 8P8C з вбудованим трансформатором (J0011D21BNL)

В даному випадку на прототип встановлюється лише один Ethernet роз’єм, необхідний для прошивки. Оскільки сам модуль має 5 інтерфейсів Ethernet (4 LAN, 1 WAN), за необхідністю можна встановити до п’яти роз’ємів. Це може знадобитися про прокладанні кабельних з’єднань — Yggdrasil байдуже, яка технологія використовується на канальному та фізичному рівнях, головне підтримка протоколу IP на мережевому рівні.

Слід зазначити, що в OpenWRT п’ятий Ethernet інтерфейс за замовчанням призначено як зовнішній (WAN). Для його використання з Yggdrasil та автоматичним взаємним виявленням сусідів у налаштуваннях OpenWRT необхідно дозволити проходження вхідного трафіку через фаєрвол, або перенести цей інтерфейс у зону «LAN». Така конфігурація є опціональною і у даній реалізації не використовується.

Рисунок 3.8 - USB-C роз’єм

Для живлення використано роз’єм USB Type-C, оскільки цей стандарт стає все більш розповсюдженим для таких цілей, замінюючи стандарт microUSB. USB-C дозволяє живити пристрій за допомогою звичайних USB-C кабелів та відповідних USB блоків живлення або павербанків, полегшуючи заміну джерела живлення за необхідністю та роблячи можливою мобільність і автономність вузлів. Слід зауважити, що попри використання USB, живлення пристрою від USB порту комп'ютера не є можливим, оскільки цей інтерфейс підкорюється специфікації USB 2.0 та обмеженню потужності у 2.5 Вт (500 мА на 5 В), тоді як для HLK-7628N необхідно як мінімум 2.64 Вт (800 мА на 3.3 В). На блоки живлення, зарядні пристрої та павербанки це обмеження не поширюється, але слід звертати увагу на максимальний струм, вказаний для конкретного джерела живлення.

Контакти «D+» та «D-» USB-C роз’єму залишаються непідключеними, оскільки обмін даними через нього не передбачається. Так як на модулі HLK-7628N реалізовано лише хост-інтерфейс USB 2.0, одночасне живлення пристрою та

зв’язок з ним через USB-C неможливі. Для такого функціоналу необхідний окремий роз’єм USB-A, під’єднаний до вищезгаданого інтерфейсу. USB інтерфейс може бути корисним для підключення додаткового бездротового адаптеру, або для кабельного підключення вузла до телефону у якості своєрідного «шлюзу» до mesh мережі. Останнє можливо при роботі телефона у режимі USB-модему.

У даній роботі функціонал інтерфейсу USB не розглядається за межами живлення пристрою через роз’єм USB-C.

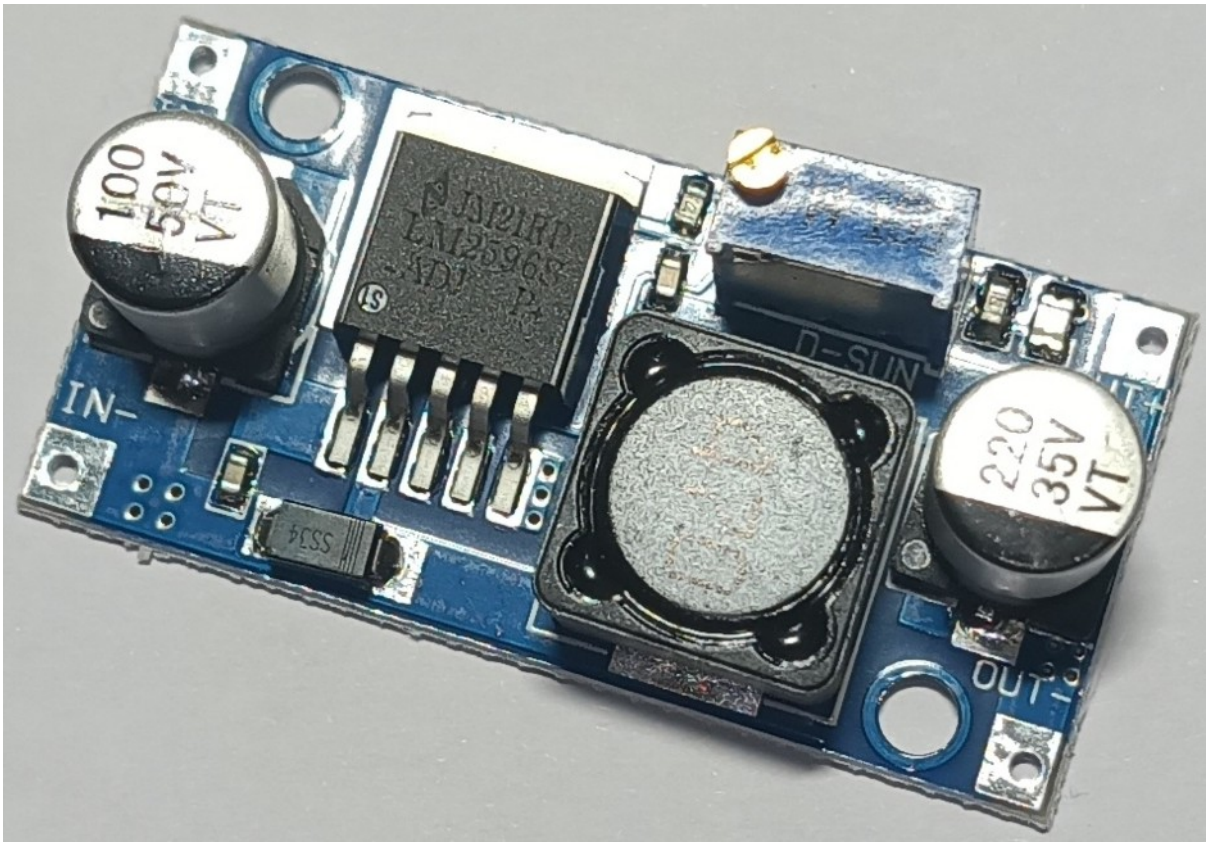


Рисунок 3.9 - DC-DC перетворювач (LM2596)

Оскільки номінальна напруга живлення модулю складає 3.3 В, а з роз’єму USB-C отримують 5 В, знадобиться знижуючий DC-DC перетворювач. Використання спеціальних блоків живлення на 3.3 В у даному випадку небажане через їх меншу розповсюдженість, більш значні втрати напруги у кабелі, а також необхідність встановлення нестандартного роз’єму живлення. До того ж, при потенційному підключенні до модуля USB-A роз’єму та використанні хост-

інтерфейсу USB все одно знадобляться стандартні для USB 5 В для живлення підключеного пристрою.

У якості DC-DC перетворювача обрано такий, який можливо налаштувати на видачу потрібної напруги за допомогою перемінного резистору. Додатково у ньому реалізовано захист від короткого замикання.

Рисунок 3.10 - USB-UART адаптер на мікросхемі PL2303HX

USB-UART адаптер загалом підійде будь-який, але більшість моделей на ринку оперує сигналами TX/RX на рівні 5 В, тоді як HLK-7628N очікує та видає сигнали на рівні 3.3 В. В такому випадку зручно використати саме адаптер на основі мікросхеми PL2303HX, що дозволяє певну модифікацію для вирішення цієї проблеми, яку описано у розділі 4.

Беручи до уваги вартість усіх комплектуючих, на даний момент собівартість одного вузла складає близько 750-800 грн. + USB-UART адаптер вартістю 40 грн.

4 РЕАЛІЗАЦІЯ ПРОТОТИПУ ВУЗЛА МЕРЕЖІ

4.1 Підготовка апаратної частини

Пристрій збирається за принциповою електричною схемою, наведеною на рисунку 4.1. Для інтерфейсу Ethernet використовується роз’єм з вбудованим трансформатором (рис. 4.2). Така електромагнітна розв’язка потрібна для зменшення впливу перешкод і захисту мікросхем модуля від надмірних збуджень напруги. Для коректної її роботи середні точки обмотки трансформатору (central taps) необхідно з'єднати з "землею" схеми через конденсатор номіналом у 0.1 мкФ.

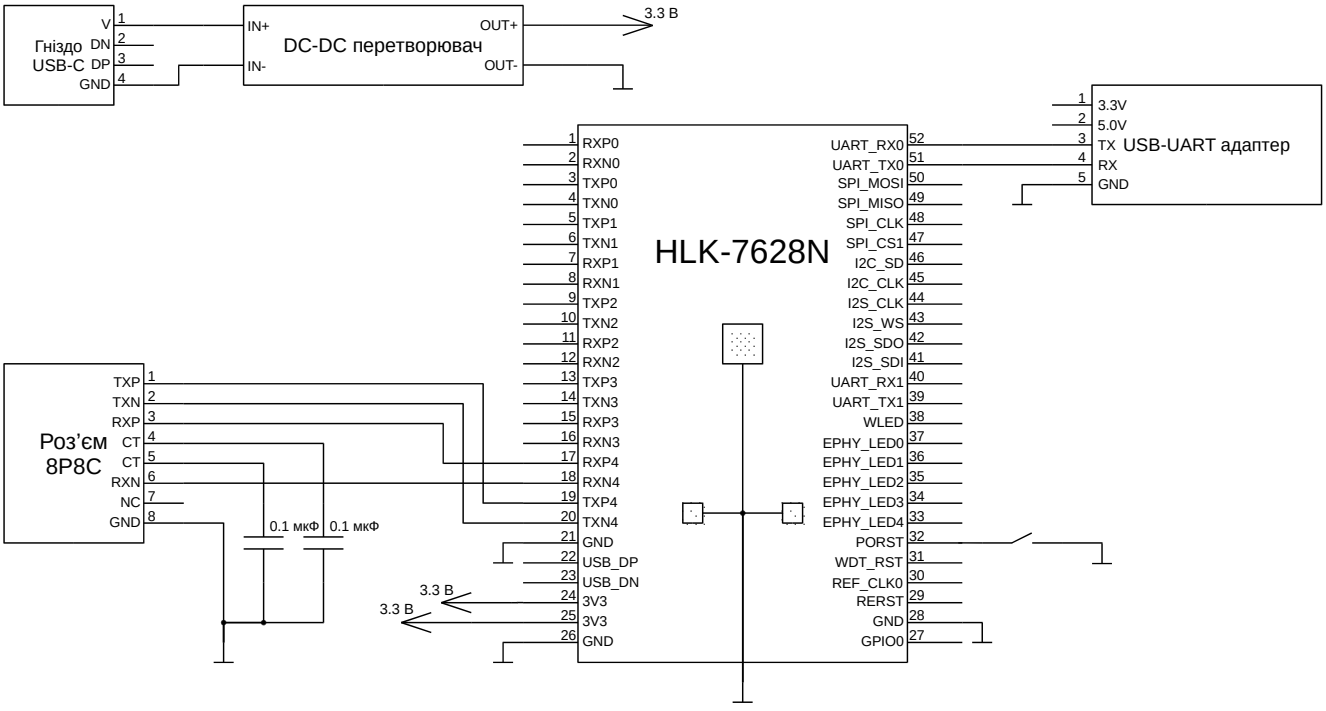


Рисунок 4.1 - Принципова електрична схема прототипу вузла

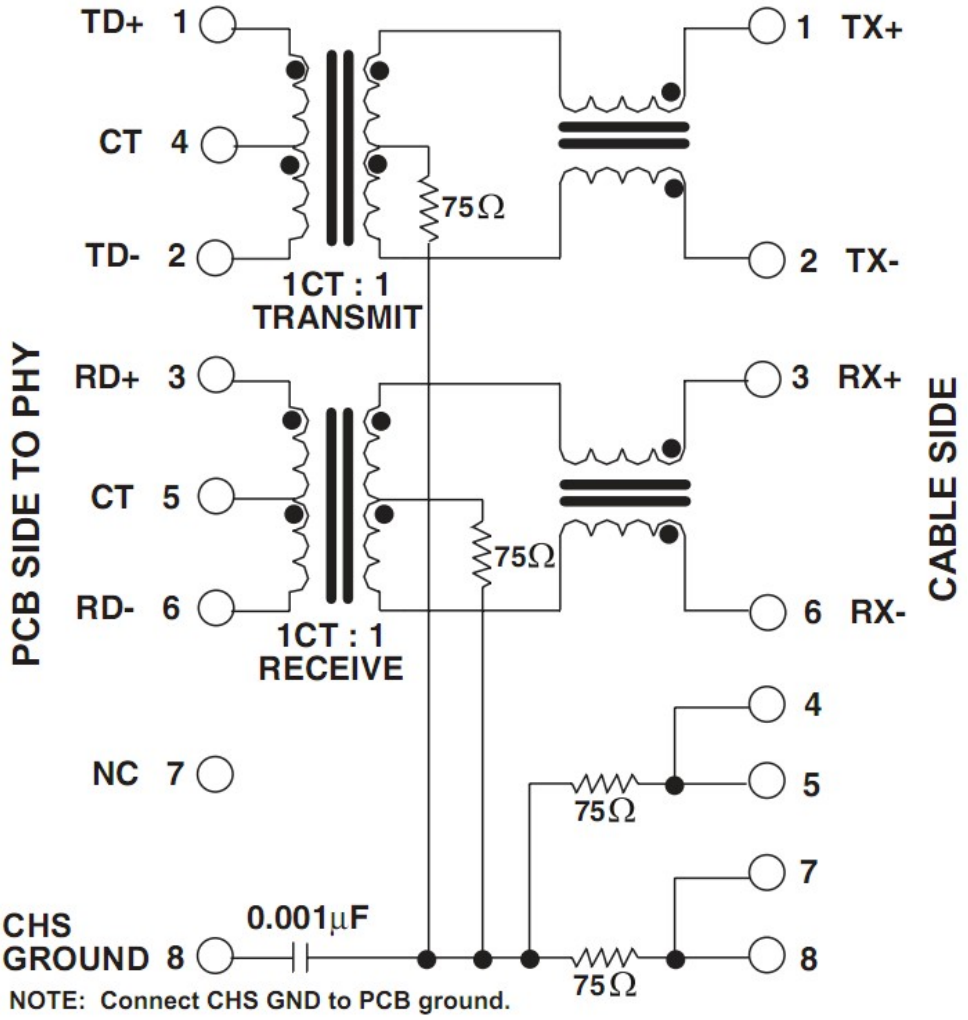


Рисунок 4.2 - Принципова електрична схема Ethernet роз'єму 8P8C з трансформатором

Живлення (3.3 В та земля) під'єднується до модуля за допомогою окремих ліній від макетної плати для кожного контакту та паяється (рис. 4.3) (для алюмінієвих жил та контактів може знадобитися активний флюс), інакше можливий брак потужності. На модулі квадратами позначено тепловідвідні майданчики, які згідно документації до модуля слід заземляти, інакше Ethernet працюватиме некоректно. Для з'єднання USB-C роз'єму з DC-DC перетворювачем також застосовується пайка. Для з'єднання з макетною платою необхідно припаяти джампери (по два на контакт) (рис. 4.4), і аналогічно це робиться для Ethernet роз'єму (рис. 4.5).

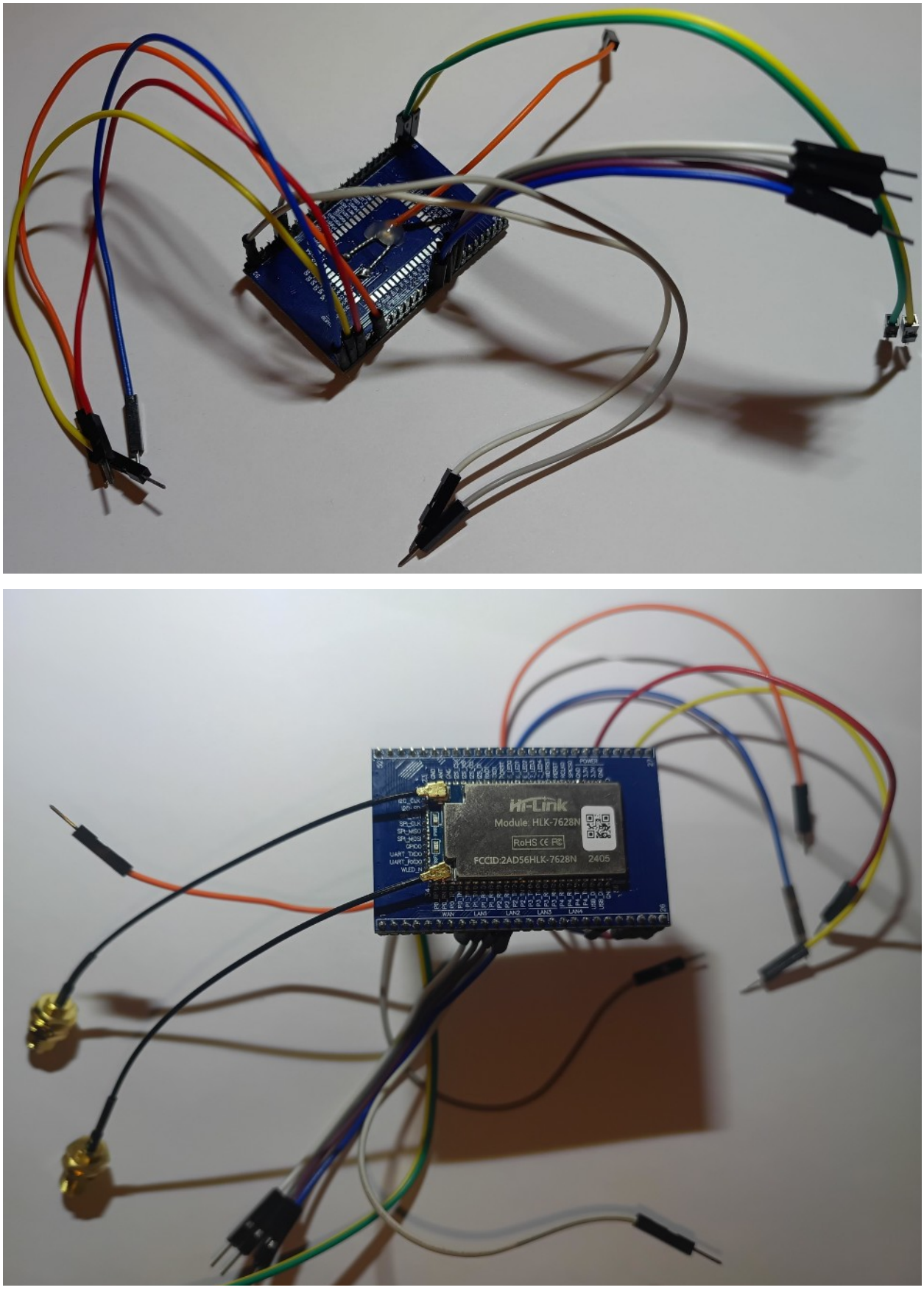


Рисунок 4.3 - Модуль HLK-7628N з під’єднаними джамперами

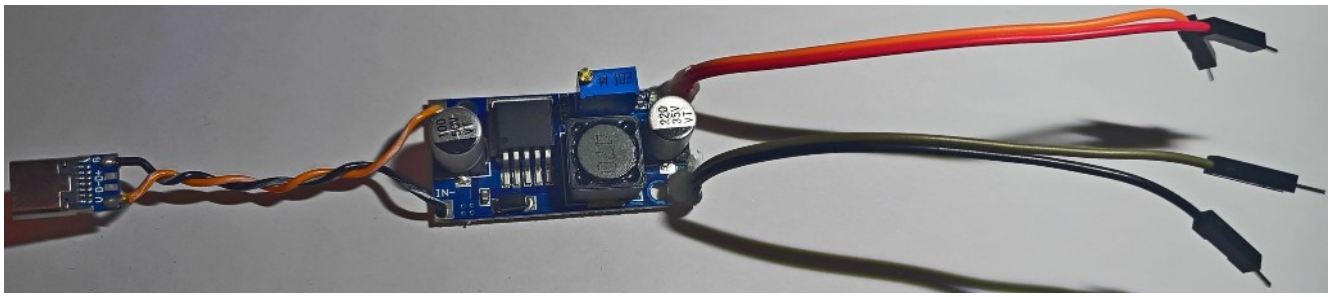


Рисунок 4.4 - DC-DC перетворювач з під’єднаними джамперами та USB-C роз’ємом

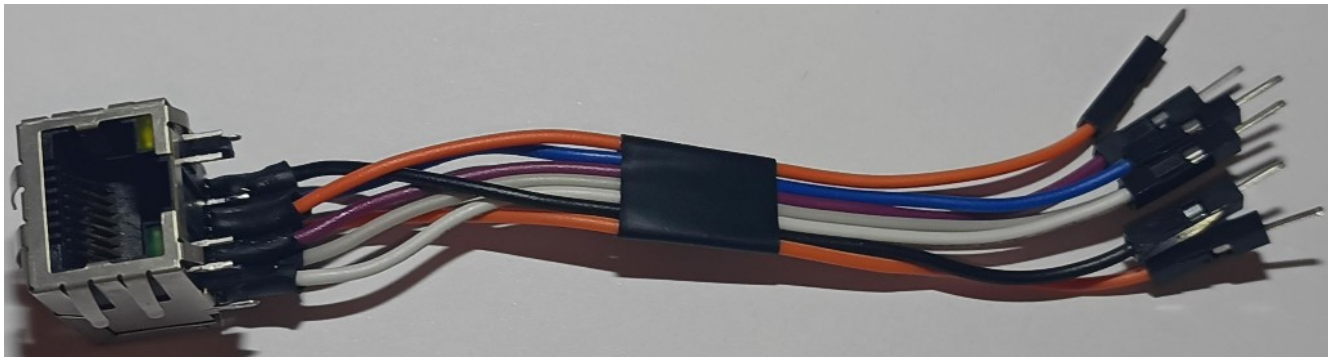


Рисунок 4.5 – Ethernet роз’єм 8P8C з під’єднаними джамперами

Підкручуванням гвинта перемінного резистору DC-DC перетворювач налаштовується на видачу напруги у 3.5 В. Таке значення підібрано виходячи з втрат напруги при з'єднанні через макетну плату та "просіданнями" під навантаженням.

Оскільки USB-UART адаптер оперує 5-вольтовою логікою, а модуль HLK-7628N 3.3-вольтовою, для їх взаємодії адаптер необхідно модифікувати. У випадку адаптера на основі мікросхеми PL2303HX напруга на сигнальних лініях регулюється напругою на четвертому виводі (VDD_232) (рис. 4.6), тобто достатньо відпаяти вивід від плати та з’єднати його об’ємним провідником із вже наявним на ній виводом 3.3 В (рис. 4.7).

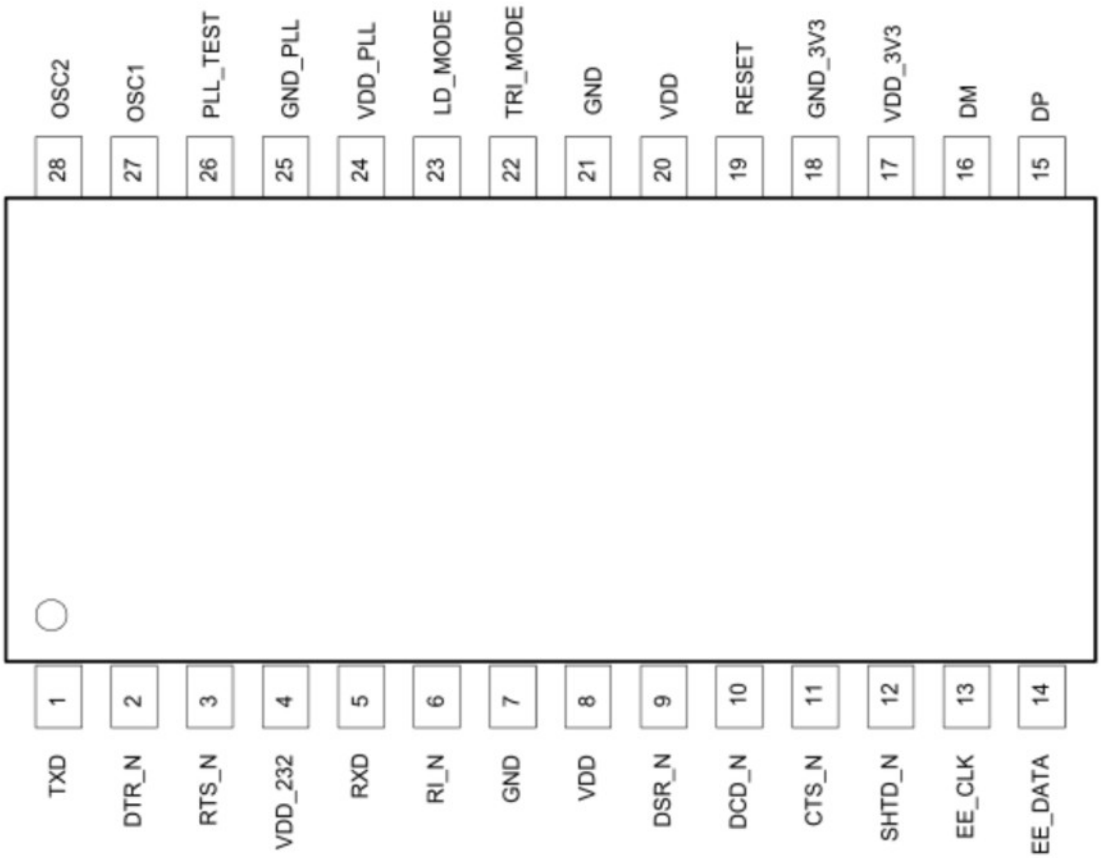


Рисунок 4.6 - Цоколівка мікросхеми PL2303HX

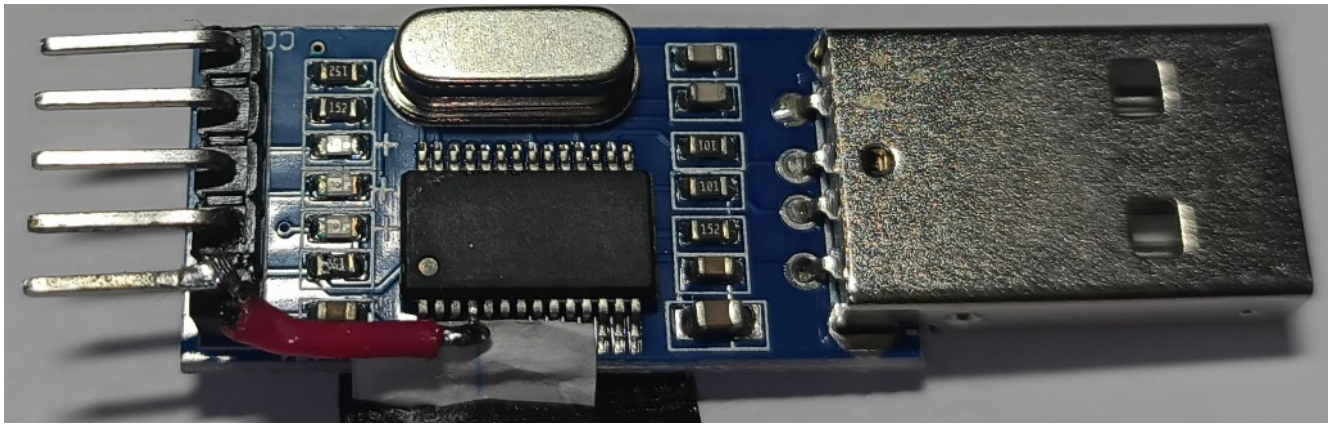


Рисунок 4.7 - Модифікований USB-UART адаптер

До U.FL роз’ємів модуля під’єднуються перехідні кабелі, на які встановлюються антени. Вмикати модуль без антен не можна, оскільки це може призвести до пошкодження бездротового інтерфейсу. Для зручності все збирається у пластмасовому контейнері. Зібраний пристрій наведено на рисунку 4.8.

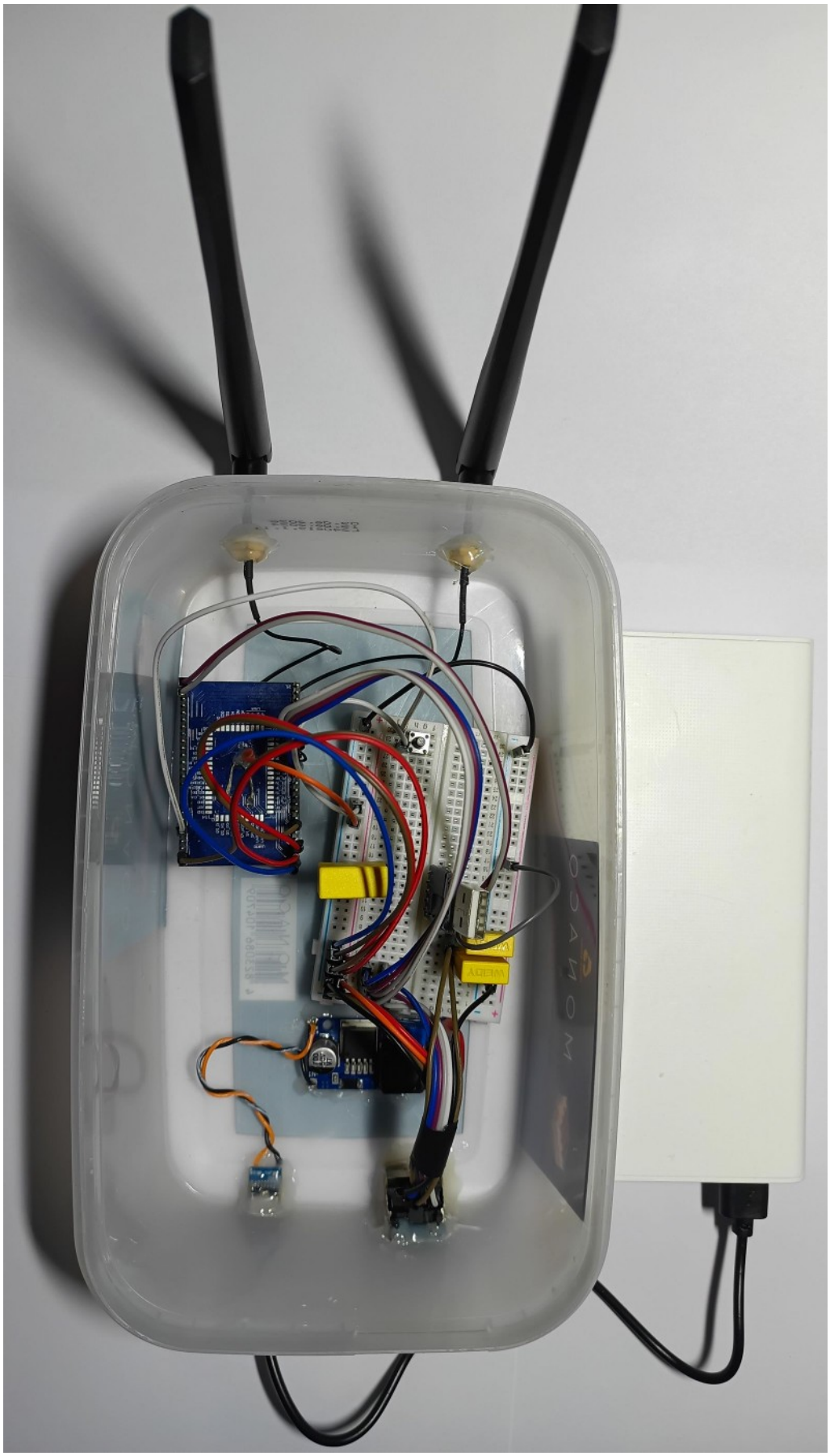


Рисунок 4.8 - Зібраний прототип вузла з живленням від павербанку

Пристрій вмикається подачею живлення на USB-C роз’єм (землю USB-UART інтерфейсу від’єднано від пристрою у момент подачі живлення). Успішний запуск підтверджується активністю світлодіоду «Wi-Fi» на модулі та самим процесом завантаження (рис. 4.9), доступним через термінал послідовного порту USB-UART адаптера (швидкість обміну — 57600 бод).

```
rflash manufacture id: ef, device id 40 19
info id : ef info->jedec_id :40160000 buf[1]:40 buf[2]:19
info id : ef info->jedec_id :40170000 buf[1]:40 buf[2]:19
info id : ef info->jedec_id :40180000 buf[1]:40 buf[2]:19
info id : ef info->jedec_id :40190000 buf[1]:40 buf[2]:19
find flash: W25Q256FV
============================================
Ralink UBoot Version: 1.0
--------------------------------------------
ASIC 7628_MP (Port5<->None)
DRAM component: 1024 Mbits DDR, width 16
DRAM bus: 16 bit
Total memory: 128 MBytes
Flash component: SPI Flash
Date:May 23 2020  Time:14:42:28
============================================
icache: sets:512, ways:4, linesz:32 ,total:65536
dcache: sets:256, ways:4, linesz:32 ,total:32768
RESET MT7628 PHY!!!!!!


  _    _   _         _  __  ______      __    ___     ___    _   _
 | |  | | | |       | |/ / |____  |    / /   |__ \   / _ \  | \ | |
 | |__| | | |       | ' /       / /    / /_      ) | | (_) | |  \| |
 |  __  | | |       |  <       / /    | '_ \    / /   > _ <  | . ` |
 | |  | | | |____   | . \     / /     | (_) |  / /_  | (_) | | |\  |
 |_|  |_| |______|  |_|\_\   /_/       \___/  |____|  \___/  |_| \_|
-------------------------------------------------------------------------
             https://github.com/gnubee-git
-------------7628N build May 23 2020 14:42:28--------------------
-------------------------------------------------------------------------

Please choose the operation:
   1: Load system code to SDRAM via TFTP.
   2: Load system code then write to Flash via TFTP.
   3: Boot system code via Flash (default).
   4: Entr boot command line interface.
   5: Load system code then write to Flash via Httpd.
   7: Load Boot Loader code then write to Flash via Serial.
   9: Load Boot Loader code then write to Flash via TFTP.
 0

3: System Boot system code via Flash.
## Booting image at bc050000 ...
   Image Name:   MIPS OpenWrt Linux-5.15.162
   Image Type:   MIPS Linux Kernel Image (lzma compressed)
```

Рисунок 4.9 - Процес запуску модуля

Прототип готовий до прошивки та подальшої роботи. Аналогічним чином збирається другий пристрій. USB-UART адаптер використовуємо той самий, оскільки він потрібен лише для першого налаштування та налагодження.

4.2 Підготовка прошивки

OpenWRT надає два варіанти створення кастомної прошивки: «Build system» та «Image Builder». У першому випадку образ системи збирається з вихідного коду, що дозволяє оптимізувати його на досить глибокому рівні, але займає доволі багато часу. Натомість, Image Builder збирає систему з готових двійкових пакетів, дозволяючи включити в образ додаткові файли, зокрема скрипти uci-defaults. Сам процес збірки прошивки при цьому займає не більше хвилини. Такого функціоналу цілком вистачить для створення прототипу вузла мережі.

Архів з Image Builder для конкретної версії OpenWRT, архітектури та моделі мікропроцесору завантажується з репозитарію OpenWRT (https://downloads.openwrt.org/). В даному випадку необхідна остання версія OpenWRT (23.05.5) для процесорів MT76X8 архітектури RAMIPS, доступна за посиланням https://downloads.openwrt.org/releases/23.05.5/targets/ramips/mt76x8/. Архів з Image Builder можна знайти внизу сторінки (openwrt-imagebuilder-23.05.5-ramips-mt76x8.Linux-x86_64.tar.xz).

Примітка: подальші дії повинні виконуватись на дистрибутиві Linux (автор використовує Parabola Linux). Додатково у системі мають бути встановлені необхідні інструменти, від яких залежить Image Builder [22].

Завантажений файл розархівується командою «tar xvf openwrt-imagebuilder-23.05.5-ramips-mt76x8.Linux-x86_64.tar.xz». Настуні дії виконуються у отриманій після цього директорії.

У прошивку необхідно включити скрипт автоматичного налаштування пристрою. Для цього створюється директорія «files», яка відображає корінь файлової системи пристрою. Файли з неї будуть включені до прошивки відповідно до створеної ієрархії, тому далі у цій директорії створюються піддиректорії «etc», «etc/uci-defaults» та файл «etc/uci-defaults/90-yggdramesh». У цей файл і записується скрипт налаштування.

Скрипт складається з наступних секцій:

- перевірка повторного запуску;
- налаштування Yggdrasil;
- налаштування бездротових інтерфейсів;
- додавання зон фаєрволу;
- додавання правил фаєрволу.

Усі налаштування робляться через командний інтерфейс UCI.

Спочатку необхідно перевірити чи запускався скрипт раніше, і якщо це так, не продовжувати виконання. Це потрібно для збереження адреси Yggdrasil після оновлення системи зі збереженням конфігурації. Програмний код наведено у лістингу 4.1.

Лістинг 4.1 - Перевірка повторного запуску скрипту

```
# Check whether the script is running after an upgrade and already
have everything set up
if [ "$(uci get network.yggdrasil)" == "interface" ]
then
    exit
fi
```

Далі налаштовується Yggdrasil, для чого його демоном генерується конфіг файл, з якого витягуються закритий та відкритий ключі. Далі через UCI у конфіг «network» додаються нові інтерфейси: «yggdrasil» для взаємодії з мережею Yggdrasil та «yggdramesh» для взаємодії з вузлами поблизу через 802.11s. Першому

вказуються отримані відкритий та закритий ключі та вимикається делегація IPv6 префіксів.

Для інших налаштувань Yggdrasil створюється окрема секція, у якій вказується надсилання та прослуховування «маяків», а також список інтерфейсів, на яких це виконується. Оскільки секція додається динамічно, UCI генерує для неї ідентифікатор, який заноситься у змінну і використовується для взаємодії з нею.

Програмний код наведено у лістингу 4.2.

Лістинг 4.2 - Налаштування Yggdrasil

```
# Set up Yggdrasil
yggdrasil -genconf -json > /tmp/yggdramesh.conf
PRIVATE_KEY="$(awk -F \" '/PrivateKey/ {print $4; exit}'
/tmp/yggdramesh.conf)"
PUBLIC_KEY="$(yggdrasil -useconffile /tmp/yggdramesh.conf
-publickey)"
uci -q batch <<-EOF
     set network.yggdrasil=interface
     set network.yggdrasil.proto='yggdrasil'
     set network.yggdrasil.private_key=${PRIVATE_KEY}
     set network.yggdrasil.public_key=${PUBLIC_KEY}
     set network.yggdrasil.delegate='0'
     set network.yggdramesh=interface
     set network.yggdramesh.proto='none'
EOF
CFG="$(uci add network yggdrasil_yggdrasil_interface)"
uci -q batch <<-EOF
     set network.${CFG}.beacon='1'
     set network.${CFG}.listen='1'
     add_list network.${CFG}.interface='br-lan'
     add_list network.${CFG}.interface='phy0-mesh0'
EOF
```

Для налаштування бездротових інтерфейсів спочатку у секції «radio0» конфігу «wireless» вказується канал (11), діапазон частот (2.4 ГГц) та ширина каналу (40 МГц). Далі Wi-Fi мережі за-замовчанням вказується SSID «Yggdramesh», додається інтерфейс «wifinet1» у режимі mesh з ID «yggdramesh» з

вимкненою маршрутизацією. Останній прив’язується до інтерфейсу «yggdramesh», створеному у лістингу 4.2.

Програмний код наведено у лістингу 4.3.

Лістинг 4.3 - Налаштування бездротових інтерфейсів

```
# Set up wireless interfaces
uci -q batch <<-EOF
     set wireless.radio0.channel='11'
     set wireless.radio0.band='2g'
     set wireless.radio0.htmode='HT40'
     set wireless.radio0.disabled='0'
     set wireless.default_radio0.ssid='Yggdramesh'
     set wireless.default_radio0.encryption='psk2'
     set wireless.default_radio0.key='testpass'
     set wireless.wifinet1=wifi-iface
     set wireless.wifinet1.device='radio0'
     set wireless.wifinet1.mode='mesh'
     set wireless.wifinet1.encryption='none'
     set wireless.wifinet1.mesh_id='yggdramesh'
     set wireless.wifinet1.mesh_fwding='0'
     set wireless.wifinet1.mesh_rssi_threshold='0'
     set wireless.wifinet1.network='yggdramesh'
EOF
```

У OpenWRT зони фаєрволу є логічними угрупуваннями мережевих інтерфейсів, трафік між якими регулюється відповідними правилами фаєрволу. У конфігу «firewall» необхідно створити дві зони для інтерфейсів «yggdrasil» та «yggdramesh» відповідно. Створеним зонам дозволяється вхідний та вихідний трафік.

Програмний код наведено у лістингу 4.4.

Далі додаються чотири правила. Перше забороняє вхідний трафік у зоні «yggdrasil» до портів 22, 80 та 443 для усіх адрес Yggdrasil, окрім певної. Це робиться для віддаленого керування вузлом через Yggdrasil з використанням особливостей адресації мережі. Білий список адрес можна за необхідності розширити.

Програмний код наведено у лістингу 4.5.

Лістинг 4.4 - Додавання зон фаєрволу

```
# Set up firewall zones
CFG=$(uci add firewall zone)
uci -q batch <<-EOF
     set firewall.${CFG}.name='yggdrasil'
     set firewall.${CFG}.input='ACCEPT'
     set firewall.${CFG}.output='ACCEPT'
     set firewall.${CFG}.forward='REJECT'
     add_list firewall.${CFG}.network='yggdrasil'
EOF
CFG=$(uci add firewall zone)
uci -q batch <<-EOF
     set firewall.${CFG}.name='yggdramesh'
     set firewall.${CFG}.input='ACCEPT'
     set firewall.${CFG}.output='ACCEPT'
     set firewall.${CFG}.forward='REJECT'
     add_list firewall.${CFG}.network='yggdramesh'
EOF
```

Лістинг 4.5 - Додавання білого списку адрес

```
# Set up firewall rules
CFG=$(uci add firewall rule)
uci -q batch <<-EOF
     set firewall.${CFG}.name='AdminWhitelist-Yggdrasil'
     set firewall.${CFG}.family='ipv6'
     set firewall.${CFG}.src='yggdrasil'
     set firewall.${CFG}.dest_port='22 80 443'
     set firewall.${CFG}.target='DROP'
     add_list firewall.${CFG}.src_ip='!\
200:b0d7:6659:cc7b:fe6c:9984:8785:74cf'
EOF
```

Третє правило працює аналогічно першому, але без білого списку адрес. Воно призначено для загальної заборони адміністративного трафіку з зони «yggdramesh».

Друге та четверте правила дозволяють весь інший трафік у зонах «yggdrasil» та «yggdramesh» відповідно. Для «yggdramesh» дозволяються лише протоколи TCP,

UDP та ICMP, оскільки лише їх використання передбачається на цьому рівні. Програмний код наведено у лістингу 4.6.

Прошивка збирається командою, наведеною у лістингу 4.7. Команда задає використання профілю «hilink_hlk-7628n», включає до образу пакети «yggdrasil», «luci», «luci-proto-yggdrasil» та «iperf3», а також файли з директорії «files». Для зручності команду можна помістити у окремий скрипт.

Лістинг 4.6 - Додавання правил фаєрволу

```
CFG=$(uci add firewall rule)
uci -q batch <<-EOF
     set firewall.${CFG}.name='Allow-Yggdrasil'
     add_list firewall.${CFG}.proto='all'
     set firewall.${CFG}.src='yggdrasil'
     set firewall.${CFG}.target='ACCEPT'
EOF
CFG=$(uci add firewall rule)
uci -q batch <<-EOF
     set firewall.${CFG}.name='Deny-Yggdramesh'
     add_list firewall.${CFG}.proto='tcp'
     add_list firewall.${CFG}.proto='udp'
     set firewall.${CFG}.src='yggdramesh'
     set firewall.${CFG}.dest_port='22 80 443'
     set firewall.${CFG}.target='DROP'
EOF
CFG=$(uci add firewall rule)
uci -q batch <<-EOF
     set firewall.${CFG}.name='Allow-Yggdramesh'
     add_list firewall.${CFG}.proto='tcp'
     add_list firewall.${CFG}.proto='udp'
     add_list firewall.${CFG}.proto='icmp'
     set firewall.${CFG}.src='yggdramesh'
     set firewall.${CFG}.target='ACCEPT'
EOF
```

Лістинг 4.7 - Команда збірки прошивки

```
make image PROFILE=hilink_hlk-7628n PACKAGES="yggdrasil luci luci-
proto-yggdrasil iperf3" FILES="files"
```

4.3 Прошивка пристрою

Процес прошивки прототипу починається з підключення його до ПК за допомогою USB-UART адаптеру та Ethernet кабелю. Через перше з'єднання виконується моніторинг та керування процесом завантаження та прошивки пристрою, а через друге передається, власне, файл прошивки. Це зроблено таким чином, оскільки передача файлу прошивки через UART займає досить багато часу і є доволі складним процесом. Натомість, можна скористатися функціоналом завантажувача, а саме можливістю прошивки за протоколом TFTP (Trivial File Transfer Protocol).

На комп'ютері запускається термінал послідовного порту USB-UART адаптера на швидкості обміну 57600 бод. В даному випадку використовується утиліта minicom, команду наведено у лістингу 4.8, де "/dev/ttyUSB1" - шлях до USB-UART адаптера в системі.

Лістинг 4.8 - Команда запуску терміналу послідовного порту USB-UART адаптера

```
sudo minicom -D /dev/ttyUSb1 -b 57600
```

Мережевому адаптеру комп'ютера привласнюється IP адреса з підмережі 192.168.1.0/24 (192.168.1.200). Перед подачею живлення від макетної плати від'єднується джампер GND USB-UART адаптеру, інакше пристрій не запуститься. Після цього на пристрій подається живлення і джампер вставляється назад.

Одразу після запуску у терміналі послідовного порту завантажувач надасть вибір опцій завантаження, необхідно обрати опцію 2 ("Load system code then write to Flash via TFTP"). У разі помилки вибору пристрій скидається кнопкою PORST на макетній платі.

Перед запуском TFTP серверу для нього створюється коренева директорія, наприклад "/tmp/tftp/", у яку як "firmware.bin" копіюється файл прошивки "bin/targets/ramips/mt76x8/openwrt-23.05.5-ramips-mt76x8-hilink_hlk-7628n-squashfs-sysupgrade.bin". Після цього виконується запуск серверу, команда залежатиме від реалізації. Команду запуску tftpd-hpa наведено у лістингу 4.9. Параметри командного рядку зазначають запуск без переходу у фоновий процес, "захищений" режим роботи (з відносними шляхами) та кореневу директорію TFTP.

Лістинг 4.9 - Команда запуску tftpd-hpa

```
sudo in.tftpd -Ls /tmp/tftp/
```

Далі у терміналі послідовного порту пристрій запитує IP адреси - власну (192.168.1.1) та TFTP серверу (192.168.1.200). Необхідно ввести їх та перевірити ім'я файлу прошивки ("firmware.bin"). Після введення цих даних пристрій завантажує файл та розпочинає процес прошивки, після якого завантажується вже з новою прошивкою і стає доступним за SSH. На цьому етапі TFTP сервер можна зупинити введенням комбінації Ctrl-C.

Останнім кроком є отримання згенерованої пристроєм адреси Yggdrasil та встановлення адміністративного паролю root. Команди для цього наведено у лістингу 4.10. Перша звертається до адміністративного UNIX сокету демона маршрутизації "/tmp/yggdrasil/yggdrasil.sock" за інформацією про вузол, друга запитує новий пароль у користувача. Обидві команди виконуються через SSH, вказані параметри командного рядку необхідні для вимкнення перевірки відкритого ключа серверу та запису його у список знайомих хостів.

Лістинг 4.10 - Команди отримання адреси Yggdrasil та встановлення паролю

```
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
root@192.168.1.1 yggdrasilctl
-endpoint=unix:///tmp/yggdrasil/yggdrasil.sock getself
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
root@192.168.1.1 passwd
```

Після цього пристрій готовий до роботи. Налаштування здійснюється автоматично за рахунок виконання системою скриптів uci-defaults під час першого завантаження після прошивки. для зручності і часткової автоматизації процесу команди з лістингів 4.6 - 4.7 можна об'єднати у скрипт, до якого додати перевірку наявності файлу прошивки, створення кореневої директорії TFTP, копіювання та видалення з неї файлу прошивки, налаштування прав доступу до файлу, а також інструкції для користувача. Вміст готового скрипту наведено у лістингу 4.11. Скрипт автоматично визначає версію OpenWRT з імені робочої директорії, використовує за замовчанням адресу пристрою "192.168.1.1" та кореневу директорію TFTP "/tmp/tftp", дозволяючи при цьому їх змінити першим та другим аргументом командного рядку відповідно.

Оновлення системи можна здійснювати передаючи файл прошивки за допомогою утиліти scp та керуючи процесом через SSH. Командою з лістингу 4.12 на пристрій у файл "/tmp/openwrt.bin" копіюється файл прошивки. Для цього утиліта scp використовується з параметром "-O", який зазначає використання старого протоколу SCP (Secure Copy Protocol) замість SFTP (Secure File Transfer Protocol), який не реалізовано у сервері Dropbear, що використовується в OpenWRT. Інші параметри командного рядку збігаються з параметрами SSH у лістингу 4.10.

Далі за допомогою SSH командою sysupgrade ініціюється процес прошивки (лістинг 4.13). Параметр "-q" зазначає "тихий" режим роботи. При такому оновленні зберігається конфігурація пристрою, в тому числі його ключі та адреса Yggdrasil. Слід зауважити, що цей функціонал вимагає обережності, оскільки збережена конфігурація може виявитись несумісною з оновленим програмним забезпеченням пристрою.

Лістинг 4.11 - Скрипт прошивки пристрою

```
#!/bin/bash
HOST=${1:-"192.168.1.1"}
TFTP_DIR=${2:-"/tmp/tftp"}
VERSION="$(pwd | cut -d"-" -f3)"
FIRMWARE_FILE="bin/targets/ramips/mt76x8/openwrt-${VERSION}-ramips-
mt76x8-hilink_hlk-7628n-squashfs-sysupgrade.bin"
if not [ -f ${FIRMWARE_FILE} ]
then
    echo "Firmware file not found, you must run the build script
first."
    exit
fi
if not [ -d ${TFTP_DIR} ]
then
    mkdir ${TFTP_DIR}
fi
cp ${FIRMWARE_FILE} ${TFTP_DIR}/firmware.bin
chmod +r ${TFTP_DIR}/firmware.bin
echo -e "Connect to the device with an Ethernet cable and UART and
boot it up.\nUpon being presented with the boot options choose
option 2 (\"Load system code then write to Flash via TFTP\") and
agree with the warning.\nSet the/ device IP (${HOST}) and server IP
(this machine's address).\nMake sure the filename is
\"firmware.bin\".\nPress any key when done."
read
echo "Launching TFTP server. You will be asked for your password.
Enter Ctrl-C when the flashing is complete and the device boots up."
sudo in.tftpd -Ls ${TFTP_DIR}
rm ${TFTP_DIR}/firmware.bin
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
root@${HOST} yggdrasilctl
-endpoint=unix:///tmp/yggdrasil/yggdrasil.sock getself
echo "Please enter the root password for the device."
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
root@${HOST} passwd
echo "Done."
```

Лістинг 4.12 - Команда завантаження файлу прошивки на пристрій

```
scp -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -O
bin/targets/ramips/mt76x8/openwrt-23.05.5-ramips-mt76x8-
hilink_hlk-7628n-squashfs-sysupgrade.bin
root@192.168.1.1:/tmp/openwrt.bin
```

Лістинг 4.13 - Команда ініціації процесу оновлення

```
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
root@192.168.1.1 sysupgrade -q /tmp/openwrt.bin
```

Аналогічно процесу початкової прошивки, команди процесу оновлення також можна зібрати у скрипт. Програмний код лістингу 4.14 додатково перевіряє наявність файлу прошивки, а також доступність пристрою. За замовчанням вказується адреса пристрою "192.168.1.1", але параметром командного рядку можна вказати будь-яку (IPv6 адреси вводяться у прямокутних дужках). Таким чином можливі і віддалені оновлення через мережу Yggdrasil.

Лістинг 4.14 - Скрипт оновлення прошивки

```
#!/bin/bash
HOST=${1:-"192.168.1.1"}
VERSION="$(pwd | cut -d"-" -f3)"
FIRMWARE_FILE="bin/targets/ramips/mt76x8/openwrt-${VERSION}-ramips-
mt76x8-hilink_hlk-7628n-squashfs-sysupgrade.bin"
if not [ -f ${FIRMWARE_FILE} ]
then
    echo "Firmware file not found, you must run the build script
first."
    exit
fi
echo "Make sure the device is reachable at ${HOST}. Press any key
when done."
read
scp -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -O $
{FIRMWARE_FILE} root@${HOST}:/tmp/openwrt.bin
if not [ "$?" == "0" ]
then
    echo "Could not send the firmware, is the device reachable?"
    exit
fi
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
root@$(echo ${HOST} | tr -d []) sysupgrade -q /tmp/openwrt.bin
echo "Firmware upgrade should be in progress. The device will become
available in several minutes."
```

4.4 Тестування

Тестування проводилося у чотири етапи:

- перевірка працездатності прототипу вузла;
- вимірювання радіусу покриття окремого вузла;
- зв'язок з 2/3/4 пристроями у мережі (рис. 4.10);
- зв'язок між web-сервером та клієнтом на Android пристроях (рис. 4.11).

4.4.1 Перевірка працездатності прототипу вузла

На першому етапі перевірялась загальна працездатність пристрою шляхом простежування процесу його завантаження через послідовний порт. Після подачі живлення очікувалося виконання прототипом наступного алгоритму:

- запуск завантажувача U-Boot, початок завантаження ядра;
- завантаження OpenWRT;
- ініціалізація бездротових інтерфейсів;
- запуск демона Yggdrasil та взаємне виявлення вузлів.

Процес роботи U-Boot наведено на рисунку 4.9. Завантажувач запускає ядро та розпочинає процес запуску OpenWRT. Повне завантаження OpenWRT зазначається доступом до терміналу OpenWRT з банером та запрошенням командного рядку (лістинг 4.15).

Лістинг 4.15 - Банер та запрошення командного рядку OpenWRT

```
BusyBox v1.36.1 (2024-11-06 10:43:07 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 23.05.5, r24106-10cc5fcd00
 -----------------------------------------------------
root@OpenWrt:/#
```

Ініціалізація бездротових інтерфейсів виконується через деякий час після запуску і зазначається відповідними логами ядра, наведеними у лістингу 4.16

Лістинг 4.16 - Ініціалізація бездротових інтерфейсів

```
[   55.765978] br-lan: port 2(phy0-ap0) entered blocking state
[   55.777120] br-lan: port 2(phy0-ap0) entered disabled state
[   55.788585] device phy0-ap0 entered promiscuous mode
[   56.823941] IPv6: ADDRCONF(NETDEV_CHANGE): phy0-ap0: link becomes
ready
[   56.837448] br-lan: port 2(phy0-ap0) entered blocking state
[   56.848568] br-lan: port 2(phy0-ap0) entered forwarding state
[   57.183931] IPv6: ADDRCONF(NETDEV_CHANGE): br-lan: link becomes
ready
[   59.947832] IPv6: ADDRCONF(NETDEV_CHANGE): phy0-mesh0: link
becomes ready
```

Після цього демон Yggdrasil розпочинає розсилку та прослухування «маяків». Через декілька секунд після цього встановлюються з'єднання з вузлами поблизу, що можна побачити у логах демону Yggdrasil, наприклад, на одному з комп'ютерів (лістинг 4.17).

Лістинг 4.17 - Взаємне виявлення вузлів, лог демону Yggdramesh на одному з комп'ютерів

```
2024/11/25 21:47:39 Connected outbound:
201:9d5c:7587:43b0:eed9:a97:c655:e0f@[fe80::40d6:3cff:fed7:42a2%wlan
0]:39537, source [fe80::6670:2ff:fe0d:8683%wlan0]:46039
2024/11/25 21:47:39 Connected outbound:
206:9b28:9f:752d:7b7b:9975:d0be:3ad4@[fe80::40d6:3cff:fed6:e652%wlan
0]:42403, source [fe80::6670:2ff:fe0d:8683%wlan0]:44179
```

Пристрій функціонує успішно, повна процедура запуску від моменту подачі живлення до доступності через Yggdrasil займає приблизно 80 секунд.

4.4.2 Вимірювання радіусу покриття окремого вузла

Другий етап тестування виконувався з двома модулями HLK-7628N, один з яких був статичним, а інший рухався, поступово збільшуючи відстань між ними.

Наявність з’єднання показувало проходження пінгів між пристроями, а також інформація зі списку сусідів, який отримується командою, наведеною у лістингу 4.18. Керування мобільним вузлом виконувалося через SSH з телефону, підключеного до точки доступу даного вузла.

Лістинг 4.18 - Команда отримання списку сусідів від демону Yggdrasil

```
yggdrasilctl -endpoint=unix:///tmp/yggdrasil/yggdrasilk.sock
getpeers
```

Тестування показало радіус покриття окремого вузла у приблизно 100 (95±5) м при умовно прямій видимості.

4.4.3 Зв'язок з 2/3/4 пристроями у мережі

На третьому етапі пристрої знаходилися поруч і «бачили» один одного. Окрім модулів HLK-7628N у мережі також брали участь два комп’ютери під управлінням Linux з бездротовими адаптерами, які підтримують стандарт 802.11s. Комп’ютери налаштовуються на підключення до mesh мережі автоматично за допомогою скрипту, наведеного у лістингу 4.19. Скрипт ініціалізує бездротовий інтерфейс, вказаний як аргумент командного рядку («wlan0» за замовчанням), переводячи його у режим Mesh Point на каналі 11 зі смугою пропускання шириною у 40 МГц, під’єднує його до mesh мережі, вимикає вбудовану маршрутизацію 802.11s та запускає екземпляр демона маршрутизації Yggdrasil.

Процедура тестування передбачала запуск утиліти iperf3 у режимі сервера на першому модулі і клієнта на другому, вимірюючи пропускну здатність TCP з’єднання. Метою такого тестування є оцінка впливу сусідніх вузлів на пропускну здатність з’єднання. На рисунку 4.10 штриховими лініями позначено взаємну доступність вузлів мережі між собою (сусідські відносини).

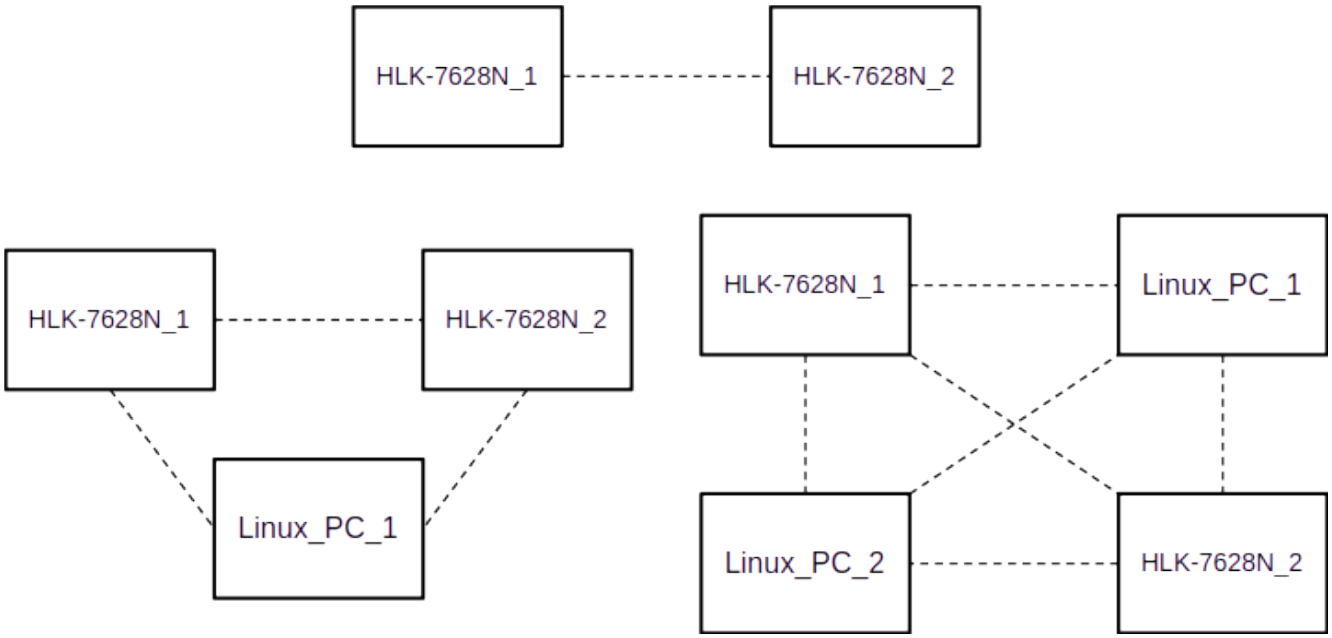


Рисунок 4.10 - Топології третього етапу тестування

Лістинг 4.19 - Скрипт налаштування комп’ютерів для підключення до mesh мережі

```
#!/bin/bash

IFNAME=${1:-"wlan0"}

echo "Disabling ${IFNAME}..."
ip link set ${IFNAME} down
echo "Setting interface type to mesh point..."
iw dev ${IFNAME} set type mp
echo "Setting channel to 11 40MHz..."
iw dev ${IFNAME} set channel 11 HT40-
echo "Enabling the interface..."
ip link set ${IFNAME} up
echo "Joining the mesh..."
iw dev ${IFNAME} mesh join yggdramesh
echo "Disabling mesh peer forwarding..."
iw dev ${IFNAME} set mesh_param mesh_fwding=0
echo "Done."
yggdrasil -useconffile /etc/yggdrasil.conf
```

Результати другого етапу тестування наведено у таблиці 4.1

Таблиця 4.1 - Результати вимірювання пропускної здатності мережі

К-ть вузлів	Макс.	Мін.	Сер.
2	3.93 Мбіт/с	3.04 Мбіт/с	3.5 Мбіт/с
3	3.53 Мбіт/с	2.04 Мбіт/с	2.91 Мбіт/с
4	3.40 Мбіт/с	1.96 Мбіт/с	2.89 Мбіт/с

З даних таблиці 3.1 зрозуміло, що зростання кількості сусідніх вузлів негативно впливає на пропускну здатність з'єднань, скоріше за все через завантаженість діапазону частот пасивним трафіком вузлів («маяками»). Такий ефект не мав би місця при використанні різних каналів для різних з'єднань, що для 802.11s з одним бездротовим інтерфейсом наразі неможливо. Слід зазначити, що тестування проводилося не в лабораторних «радіо-чистих» умовах, тому отримані значення пропускної здатності є орієнтовними.

4.4.4 Зв'язок між web-сервером та клієнтом на Android пристроях

На четвертому етапі до точок доступу модулів підключено два телефони під управлінням Android. На рисунку 4.11 штрих-пунктирними лініями позначено з'єднання з точками доступу модулів в інфраструктурному режимі Wi-Fi. На телефонах запущено офіційну імплементацію Yggdrasil для Android (рис. 4.12). На першому телефоні запущено web-сервер (пакет «apache2» додатку Termux [23]), другий під'єднується до нього через Yggdrasil за допомогою браузера. Аналогічно до web-серверу під'єднується перший комп'ютер. У разі успіху в обох випадках клієнт отримає сторінку з повідомленням.

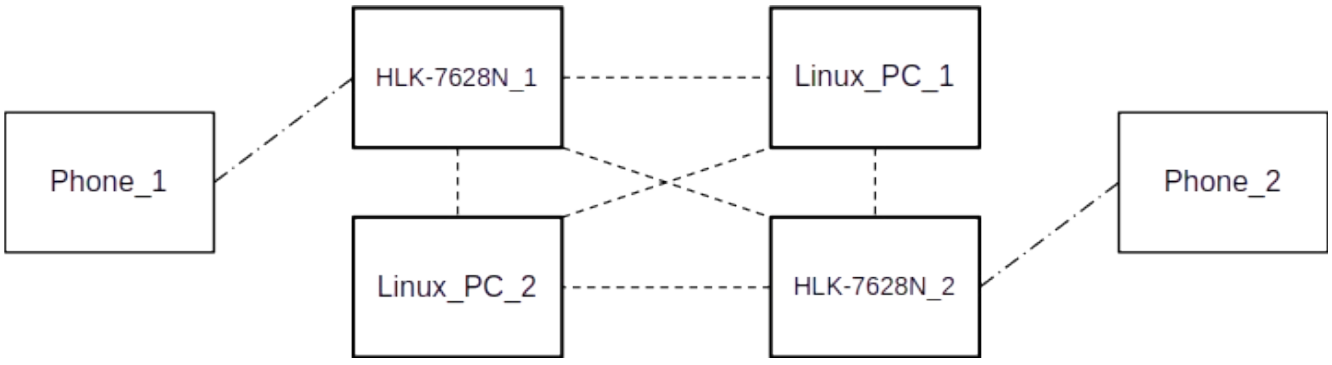


Рисунок 4.11 - Топологія третього етапу тестування

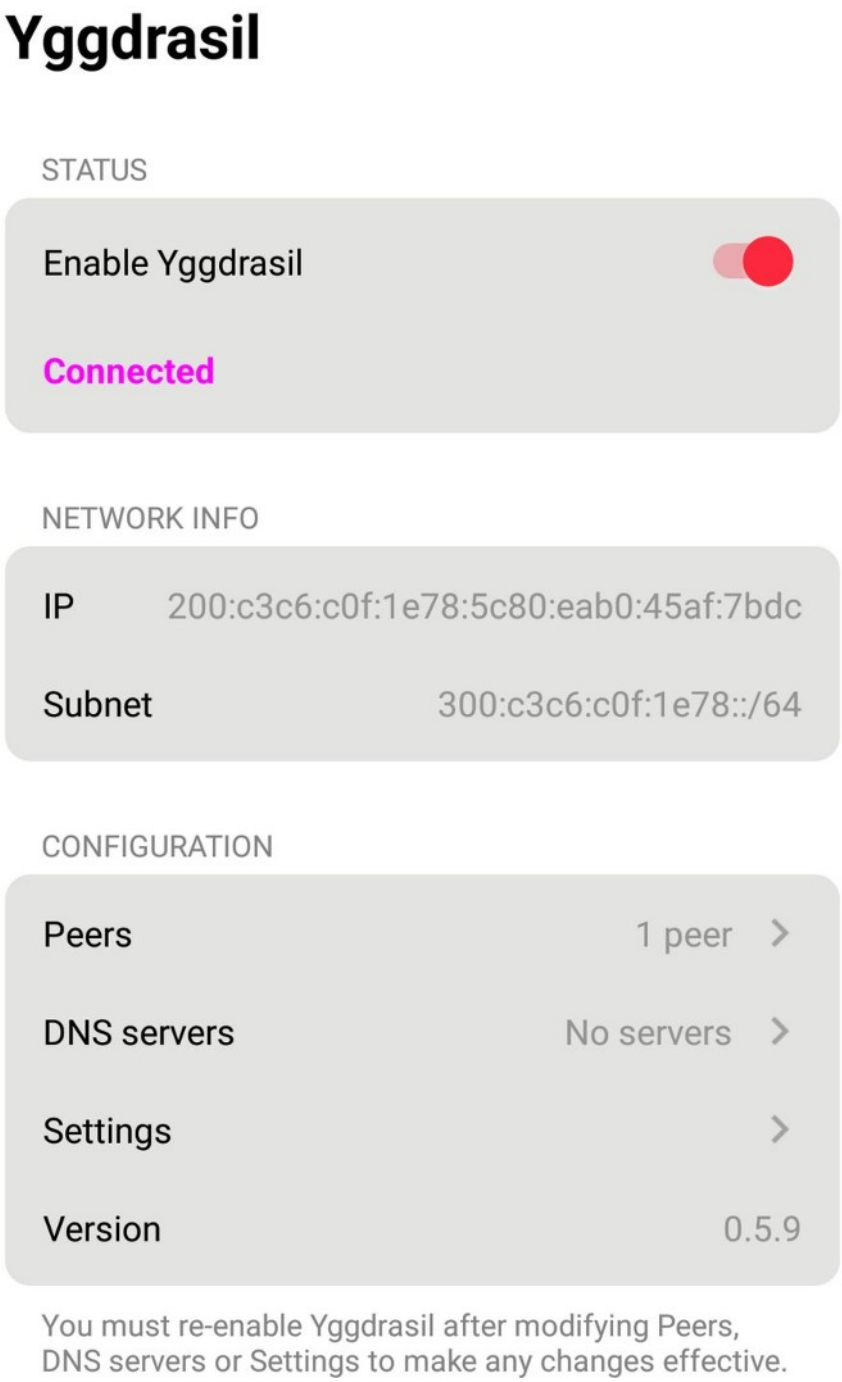


Рисунок 4.12 - Додаток Yggdrasil на першому телефоні

У адресний рядок браузеру у прямокутних дужках вводиться адреса Yggdrasil першого телефону з рисунку 4.12 та порт 8080 («http://[200:c3c6:c0f:1e78:5c80:eab0:45af:7bdc]:8080/»). Результат наведено на рисунках 4.13 та 4.14.

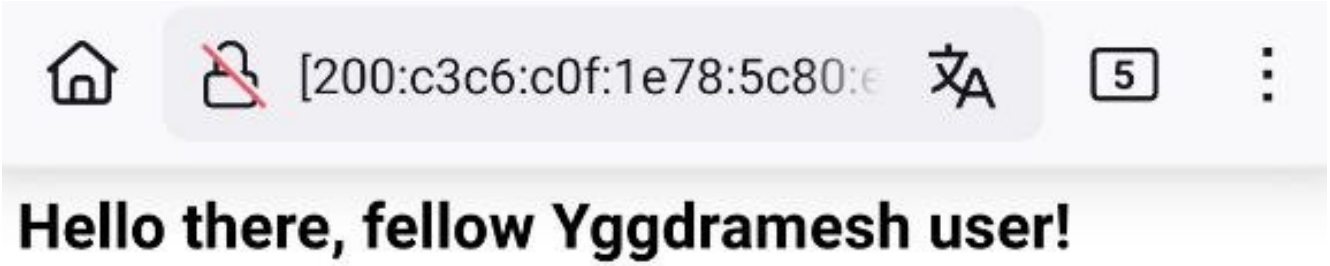


Рисунок 4.13 - Сторінка web-серверу, відкрита на другому телефоні

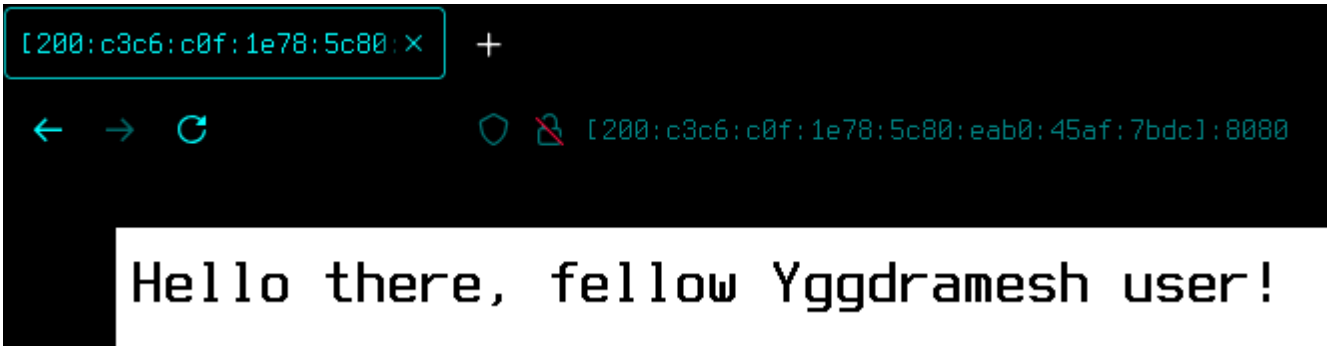


Рисунок 4.14 - Сторінка web-серверу, відкрита на комп’ютері

В обох випадках клієнт успішно звертається до серверу та отримує сторінку. Аналогічно у мережі можна запускати майже будь-які додатки, в тому числі P2P месенджери без необхідності у сервері, оскільки будь-який вузол доступний будь-якому іншому.

ВИСНОВКИ

В процесі проведення досліджень та виконання роботи визначено доцільність використання схеми маршрутизації Yggdrasil, як основи масштабної децентралізованої mesh мережі, а також проведено реалізацію прототипу вузла мережі, яка проєктується. Експерименти з демоном маршрутизації проведено на різного розміру топологіях із використанням середовищ meshnet-lab і coreemu-lab. В результаті визначено орієнтовну структуру масштабної децентралізованої mesh мережі та технічні вимоги до її вузлів. Експерименти демонструють, що Yggdrasil значно перевершує інші протоколи маршрутизації сітчастої мережі в обмеженні переходів та визначають параметри використання системних ресурсів. Тому Yggdrasil є перспективним методом розгортання великомасштабних децентралізованих сітчастих мереж незалежних вузлів і альтернативою традиційним централізованим ієрархічним мережам.

Під час тестування виявлено деякі особливості застосування Yggdrasil, як то збільшення використання процесору на довгих гілках остовного дерева і аномалії трафіку, що внесені блукаючим кореневим вузлом. Як засіб пом'якшення запропоновано попереджувальний майнінг малих ключів для детермінованого розміщення кореневих вузлів. Наведено орієнтовну модель системи та обґрунтовано використання стандарту IEEE 802.11s з вимкненою маршрутизацією як базової технології канального рівня.

Проведено тестування і створено прототип вузлу мережі на основі модуля HLK-7628N виробництва Hi-Link та ОС OpenWRT. Описано процеси збірки та прошивки прототипів. Процеси початкової прошивки та оновлення пристроїв частково автоматизовано за допомогою розроблених скриптів. Налаштування пристроїв після прошивки автоматизовано розробленим скриптом uci-defaults.

Тестування показало повну працездатність реалізованих прототипів і мережі з двох прототипів, двох пристроїв під управлінням Linux та двох телефонів під

управлінням Android. Визначено орієнтовні ліміт покриття окремого вузла та пропускну здатність мережі при даній конфігурації. Продемонстровано приклад прикладного використання мережі з web-сервером та клієнтом на Android пристроях.

Подальшими напрямками дослідження визначено удосконалення прошивки, структури і конструкції вузлів мережі з метою покращення стабільності їх роботи, розширення функціоналу та збільшення пропускної здатності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nishizawa H. Architecting Cloud-native Optical Network with Whitebox Equipment. Optical Fiber Communication Conference, м. San Diego, California. Washington, D.C., 2020. P. W3C. 5. DOI: https://doi.org/10.1364/ofc.2020.w3c.5.

2. Greig J. A. Wireless Mesh Networks as Community Hubs: Analysis of Small-Scale Wireless Mesh Networks and Community-Centered Technology Training. Journal of Information Policy. 2018. 8(1). P. 232–266. DOI: https://doi.org/10.5325/jinfopoli.8.1.0232.

3. Rudkovskyi O., Kyrychek H. Interaction support system of network aplications. CEUR Workshop Proceedings. 2020. Vol. 2832. P. 11–23.

4. Wu D., Liebeherr J. A Low-Cost Low-Power LoRa Mesh Network for Large-Scale Environmental Sensing. IEEE Internet of Things Journal. 2023. P. 1. DOI: https://doi.org/10.1109/jiot.2023.3270237.

5. Yamamoto R., Yamazaki T., Ohzahata S. VORTEX: Network-Driven Opportunistic Routing for Ad Hoc Networks. Sensors. 2023. 23(6). P. 2893. DOI: https://doi.org/10.3390/s23062893.

6. Prabu S., Maheswari M., Jothi B., Banupriya J., Bindu G. Efficient Bloom Filter-Based Routing Protocol for Scalable Mobile Networks. Engineering Proceedings. 2023. 59(1). P. 75. DOI: https://doi.org/10.3390/engproc2023059075.

7. Grandi F. On the analysis of Bloom filters. Information Processing Letters. 2018. Vol. 129. P. 35–39. DOI: https://doi.org/10.1016/j.ipl.2017.09.004/

8. Yggdrasil Network. URL: https://yggdrasil-network.github.io/ (дата звернення: 24.10.2024).

9. Пестов О., Киричек Г., Тягунова М. Схема маршрутизації yggdrasil як основа для великомасштабних децентралізованих meshмереж. Materials of the XII International Scientific Conference «Information-Management Systems and Technologies», 23th – 25th September, 2024, Odesa, P. 71–75.

10. Pestov O., Kyrychek H., Tiahunova M. Yggdrasil routing scheme as a basis for large-scale decentralized mesh networks. In Proceedings of the ICST-2024. CEUR Workshop. Vol. 3790. P. 110–122.

11. Brendel J., Cremers C., Jackson D., Zhao, M. The Provable Security of Ed25519: Theory and Practice. IEEE Symposium on Security and Privacy (SP). 2021. P. 1659-1676. DOI: https://doi.org/10.1109/sp40001.2021.00042.

12. Khaledi M., Rovira-Sugranes A., Afghah F., Razi A. On greedy routing в dynamic uav networks. IEEE International Conference on Sensing, Communication and Networking (SECON Workshops), 2018. P. 1-5. DOI: https://doi.org/10.1109/seconw.2018.8396354.

13. Messan PN, Krupinski S., Maurelli F., Vallicrosa G., Ridao, P. Evaluation of computer networking methods for interaction with remote robotic systems. IEEE AFRICON. 2021. P. 1-6. DOI: https://doi.org/10.1109/africon51333.2021.9570912.

14. Reich B. Wifi-Ad-hoc Mesh Networks for mobile Systems (Doctoral dissertation, Hochschule für Angewandte Wissenschaften Hamburg). 2024. URL: http://hdl.handle.net/20.500.12738/14753 (дата звернення: 24.10.2024).

15. Schubert D., Jaeger B., Helm M. Network Emulation using Linux Network Namespaces. Network. 2019. 57. DOI: https://doi.org/10.2313/NET-2019-10-1_11.

16. Massaron L., Mueller J. P. Python for Data Science For Dummies. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2015. 418 p. ISBN(978-1-118-84418-2).

17. Campesato O. Chapter 5: Matplotlib and Seaborn. Data Literacy with Python. 2023. P. 117–164. DOI: https://doi.org/10.1515/9781501518652-006.

18. Baumgärtner L., Meuser T., Bloessl B. coreemu-lab: An automated network emulation and evaluation environment, IEEE Global Humanitarian Technology Conference (GHTC), 2021. P. 304-311. DOI: https://doi.org/10.1109/ghtc53159.2021.9612475.

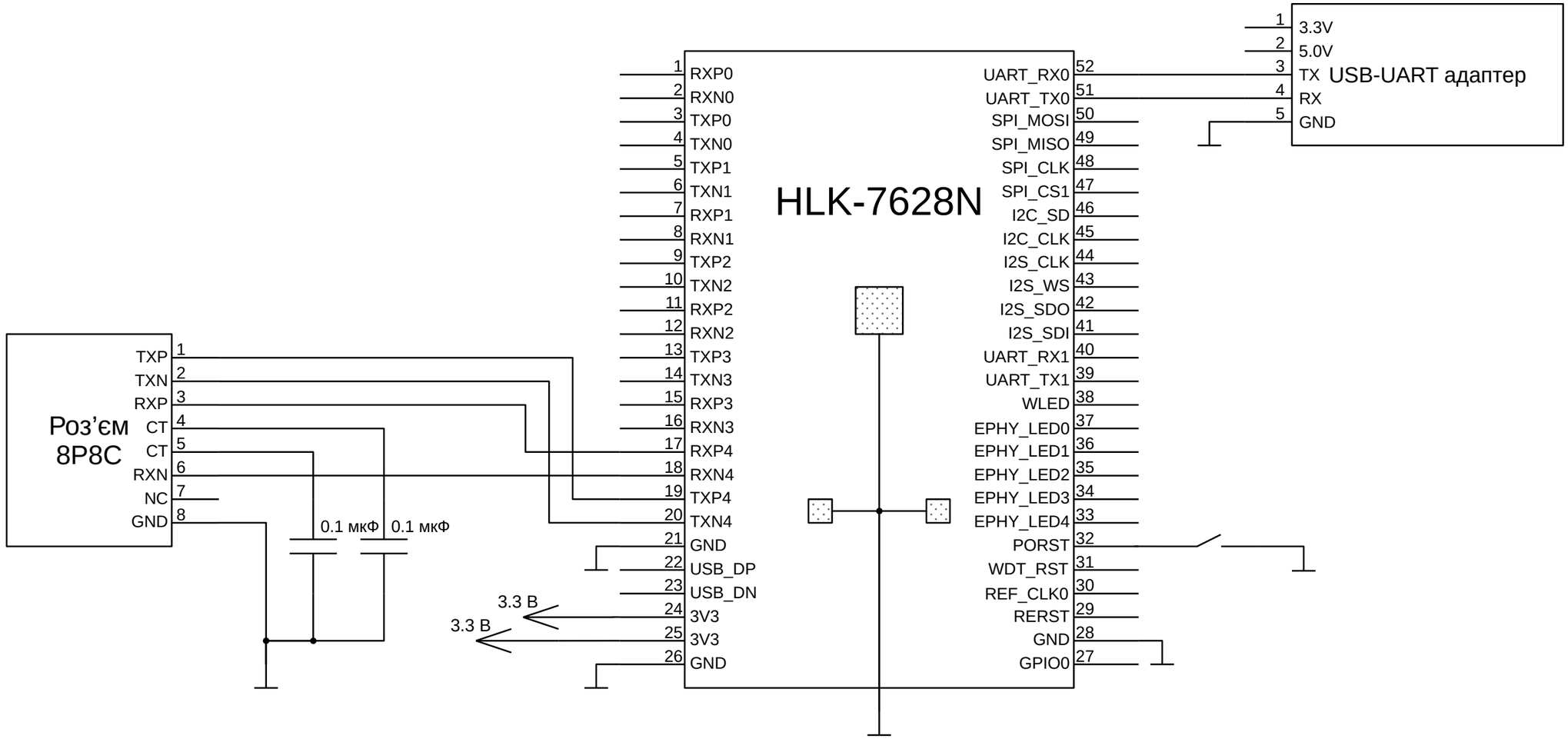
19. Ahrenholz J., Danilov C., Henderson T. R., Kim J. H. CORE: A real-time network emulator, IEEE Mil. Commun. Conf., 2008. P. 1-7. DOI: https://doi.org/10.1109/milcom.2008.4753614.

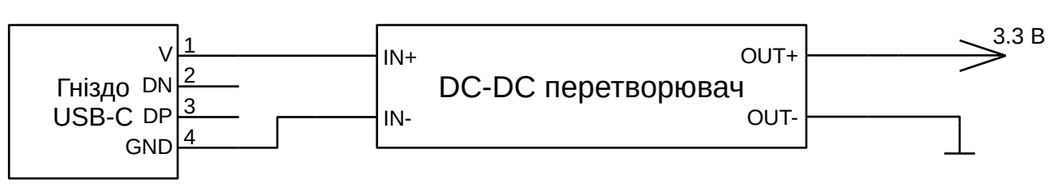
20. ps(1) - Linux manual page. URL: https://man7.org/linux/man-pages/man1/ps.1.html (дата звернення: 24.10.2024).

21. Casetti CE, Chiasserini CF, Duan Y., Giaccone P., Manriquez, AP. Data connectivity and smart group formation в Wi-Fi direct multi-group networks. IEEE transactions on network and service management, 2017, 15 (1), P. 245-259. DOI: https://doi.org/10.1109/tnsm.2017.2766124.

22. Using the Image Builder. OpenWrt Wiki. URL: https://openwrt.org/docs/guide-user/additional-software/imagebuilder (дата звернення: 17.11.2024).

23. The main Termux site and help pages. Termux. URL: https://termux.dev/en/ (дата звернення: 24.11.2024).





					13.02070849.51315 Е3	
					Масштабна децентралізована mesh мережа на основі Yggdrasil Принципова електрична схема прототипу вузла мережі	Літ. / Маса / Масштаб
Зм.	Лист.	№ докум.	Підп.	Дата		
Розробив		Пестов О.Д.				Лист. 1 / Листів 1
Перевірив		Киричек Г.Г.				
Т. контр.						НУ «Запорізька політехніка» КНТ-513м
Н. контр.		Щербак Н.В.				
Затв.		Кудерметов Р.К.				