РЕФЕРАТ

Магістерська кваліфікаційна робота містить основну частину, що складається з вступу, трьох розділів, висновку та двох додатків. Загальний обсяг роботи становить 72 сторінки, 29 ілюстрацій, 35 посилань.

Метою дипломної роботи є дослідження методів покращення цілісності передачі інформації у запропонованих системах відеоспостереження швидкого розгортання на основі геометричних алгоритмах на графах.

Дипломний проект надає докладний аналіз протоколів маршрутизації сітчастих мереж, які можуть бути використані для організації каналів зв’язку в СВШР.

Запропонована алгоритмічна структура на основі геометричних алгоритмів для покращення якості передачі даних в сітчатих самоорганізаційних мережах на основі тріангуляції Делоне, алгоритму побудови діаграми Вороного та алгоритму вимітання площини.

Результати проведеної роботи можуть бути використані при плануванні, проектуванні, оновленні та оптимізації систем відеоспостереження швидкого розгортання, безпілотних літальних апаратів відеоспостереження та передачі даних в самоорганізаційних сітчатих мережах.

Охоронні системи, СВШР, БПЛА, Mesh-мережі, Захист інформації в бездротових мережах, HWMP, геометричні алгоритми, використання графів в мережевих топологіях.

Abstract

In this work, we propose a framework using geometric algorithms to largely reduce the interference among nodes in a wireless mesh network. We first convert network problems into geometry problems in graph theory, and then solve the interference problem by geometric algorithms. We first define line intersection in a graph to reflect radio interference problem in a wireless mesh network. We then use plan sweep algorithm to find intersection lines, if any; employ Voronoi diagram algorithm to delimit the regions among nodes; use Delaunay

Triangulation algorithm to reconstruct the graph in order to minimize the interference among nodes. Finally, we use standard deviation to prune off those longer links (higher interference links) to have a further enhancement. This hybrid solution is proved to be able to significantly reduce interference in O(n log n) time. Simulations show that the proposed framework is effective in increasing throughput, reducing both packets loss rate and delay time on average.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА УМОВНИХ СКОРОЧЕНЬ

СВШР	**Система відеоспостереження швидкого розгортання**
СОП	**Система охорони периметру**
БПЛА	**Безпіло́тний літа́льний апара́т**
ДКЛА	**Дистанційно-керований літальний апарат**
ІТЗ	**Інженерно-технічні засоби**
HWMP	**Hybrid Wireless Mesh Protocol**
AODV	**Ad hoc On-demand Distance Vector**
OLSR	**Optimized Link State Routing**
BATMAN	**Better Approach To Mobile Ad hoc Networks**
DSDV	**Destination-Sequenced Distance Vector**
HSLS	**Hazy Sighted Link State**
EIGRP	**Enhanced Interior Gateway Protocol**
ZRP	**Zone Routing Protocol**
FSR	**Fish-eye State Routing Protocol**
SHWMP	**Secury HWMP**
QoS	**Quality of services**
ETX	**Expected Transmission Count**
DR	**Delivery Rate**
PDR	**Packet Delivery Rate**
OSI	**Open Systems Interconnection model**
STA	**Station Networking**
PREQ	**Path Request**
PREP	**Path Reply**
PERR	**Path Error**
Rann	**Root Announcement**
MF	**Mutable Field**
NMF	**Non Mutable Field**
TTL	**Time To Live**

IGTK	**Integrity Group Temporal Key**
IPN	**Integrity Packet Number**
PTK	**Pairwaise Transient Key**
MMIE	**MIC Integrity Element**
OGM	**Originator Message**

ВСТУП

Згідно НДТЗІ, рішення про встановлення ІІІ категорії об'єктам, на яких обробляється технічними засобами та/або озвучується інформація з обмеженим доступом, може здійснюватися за рішенням розпорядників (користувачів) інформації [1,3]. Для здійснення контролю доступу на таких об'єктах рекомендується розгортання периметральної охоронної системи.

Системи відеоспостереження являються важливим елементом периметральних охоронних систем та першим технічним рубежем захисту на об'єктах інформаційної діяльності (ОІД); надійність і ефективність цього рубежу дуже важлива для раннього виявлення порушення режиму доступу.

Актуальність роботи. Більшість охоронних систем швидкого розгортання (ОСШР), які використовуються в наш час, відноситься до сигналізаційних систем, одним з головних недоліків яких є те, що система дає нам інформацію не про те, що насправді відбувається у контрольованій зоні, а лише сам факт спрацювання датчика[2]. Причину цього спрацювання можуть надати прилади відео спостереження. Тому актуально розглянути можливість та схему використання приладів відео-спостереження для розширення функціональності ОСШР.

Мета роботи – дослідження методів покращення цілісності передачі інформації у запропонованих системах відеоспостереження швидкого розгортання на основі геометричних алгоритмах на графах.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

- Дослідити механізми захисту інформації при передачі даних в бездротових мережах.
- Сформулювати представлення мережеву топологію передачі інформації у вигляді графу.
- Запропонувати модель покращення якості передачі на основі геометричних алгоритмів.
- Дослідити та порівняти отримані результати за допомогою комп'ютерного моделювання.

Об’єкт дослідження – захист інформації на об'єктах інформаційної діяльності

Предмет дослідження – системи відеоспостереження швидкого розгортання.

РОЗДІЛ 1
ОГЛЯД КОНЦЕПЦІЇ ОХОРОННИХ СИСТЕМ ВІДЕОНАГЛЯДУ ШВИДКОГО РОЗГОРТАННЯ

1.1 Концепція системи відеоспостереження, як периметральної охоронної системи.

Сучасні електронні системи охорони вельми різноманітні і в цілому досить ефективні. Однак більшість з них мають загальний недолік: вони не можуть забезпечити раннє детектування вторгнення на територію об'єкта. Такі системи, як правило, орієнтовані на виявлення порушника, який вже проник на охоронювану територію або в будівлю. Це стосується, зокрема, систем відеоспостереження; вони часто за допомогою пристрою відеоспостереження можуть підтвердити факт вторгнення після того, як він вже відбувся. Кваліфікований порушник завжди розраховує на певне тимчасове "вікно", яке проходить від моменту проникнення на об'єкт до моменту спрацьовування сигналізації. Мінімізація цього інтервалу часу є корінним чинником, що визначає ефективність будь-якої охоронної системи, і в цьому сенсі привабливість периметральної охоронної сигналізації незаперечна.

Периметрова межа об'єкту є найкращим місцем для раннього детектування вторгнення, тому що порушник взаємодіє в першу чергу з фізичним периметром і створює збурювання, які можна зареєструвати спеціальними датчиками. За певних умов порушник може уникнути фізичного контакту з периметром. В цьому випадку можна використовувати "об'ємні" датчики вторгнення, які зазвичай грають роль вторинної лінії захисту.

Будь-яка периметрова система охорони повинна відповідати певному набору критеріїв, деякі з яких перераховані нижче:

- можливість раннього виявлення порушника – ще до його проникнення на об'єкт;
- точне проходження контурам периметра, відсутність "мертвих" зон;
- по можливості приховане встановлення датчиків системи;

- незалежність параметрів системи від сезону (зима, літо) і погодних умов (дощ, вітер, град і т.д.);

- несприйнятливість до зовнішніх чинників "нетривожного" характеру - індустріальні перешкоди, шум від транспорту, що проходить поруч, дрібні тварини і птахи;

стійкість до електромагнітних перешкод - грозові розряди, джерела потужних електромагнітних випромінювань і т.п.

Очевидно, що периметральна охоронна система повинна володіти максимально високою чутливістю, щоб виявити навіть досвідченого порушника. У той же час ця система повинна забезпечувати по можливості низьку ймовірність помилкових спрацьовувань. Причини помилкових тривог можуть бути різними. Система може, наприклад, зреагувати при появі в зоні охорони птахів або дрібних тварин. Сигнал тривоги може з'явитися при сильному вітрі, граді або дощі. Крім того, помилкова тривога може виникнути з "технологічних" причин: неграмотний монтаж датчиків на огорожі, неправильне налаштування електронних блоків або просто незадовільний інженерний стан самої огорожі, яка може, наприклад, вібрувати при сильному вітрі.

Вибір оптимального варіанта побудови СОП здійснюється на основі порівняльного аналізу їх основних характеристик, в першу чергу, таких, як ефективність і вартість.

У методах підтримки прийняття рішень ці характеристики виступають в якості основних критеріальних показників.

Практичний досвід показує, що для обгрунтованого вибору способу побудови системи охорони периметра порівняння тільки за характеристиками ефективності і вартості виявляється недостатнім і потребує розгляду ряду інших характеристик. Ці характеристики, що оцінюють конкретні властивості СОП, істотно розширюють і доповнюють уявлення про способи їх побудови на етапі прийняття рішення і, в цілому, представляються як функціональні вимоги.

У процесі узагальнення досвіду вибору способів побудови СОП при оснащенні різних типів об'єктів були визначені вимоги до СОП на доповнення до ефективності і вартості (Рис. 1.1).

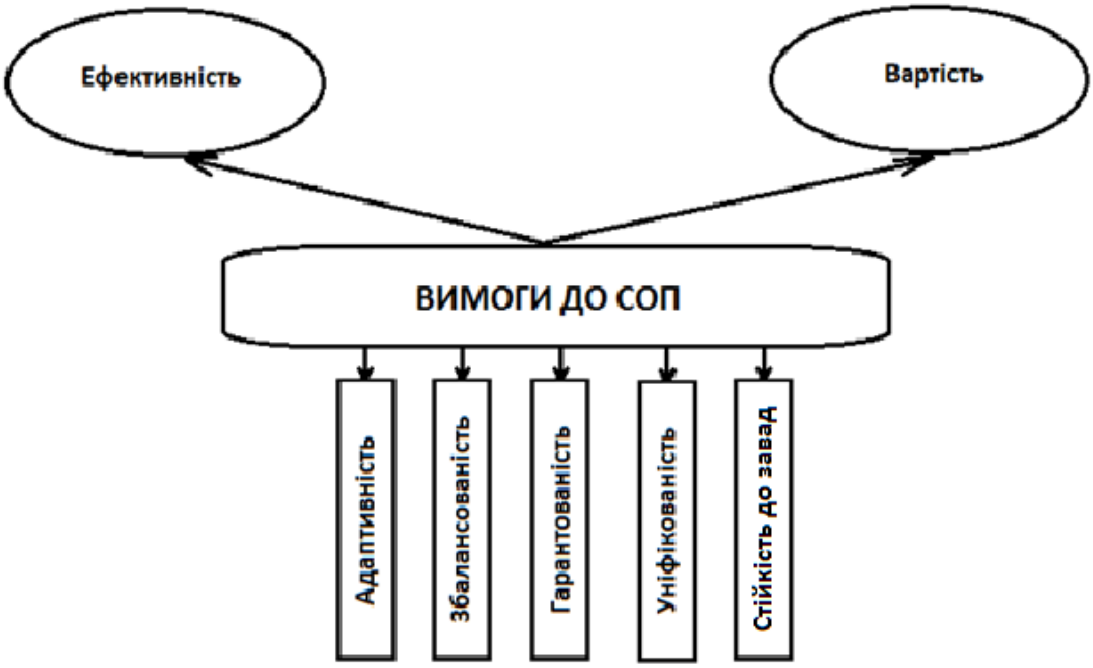


Рис. 1.1 – Класифікація вимог до системи охорони периметра

Адаптивність системи, під якою розуміється її пристосованість до установки на об'єктах різних категорій і розмірів, а також у різних фізико-географічних умовах при будь-яких рельєфах місцевості. Ця вимога пред'являється не тільки і не стільки об'єктом, скільки системою ще більш високого рівня. Під такою системою в простому випадку можна розглядати міністерство (відомство), у підпорядкуванні якого перебуває об'єкт. Формально адаптивність означає те, що інженерно-технічні засоби (ІТЗ) можуть бути встановлені на об'єктах декількох категорій, різних розмірів, що знаходяться в будь-яких зовнішніх умовах експлуатації. Неформально адаптивність означає можливість ІТЗ змінювати свою структуру, топологію, організацію функціонування і управління залежно від особливостей і вимог, що пред'являються об'єктам.

Збалансованість системи, під якою розуміється раціональна інтеграція технічних підсистем, а також підсистеми фізичних бар'єрів і сил охорони.

Збалансованість передбачає раціональність поєднання властивостей утворюючих підсистем щодо забезпечення виконання функцій виявлення, затримки і нейтралізації порушника, а також їх побудова і управління функціонуванням. За цими ознаками показник збалансованості тісно взаємопов'язаний з ефективністю. У той же час при пред'явленні найвищих вимог по ефективності їх досягнення стає можливим лише за максимальної реалізації властивостей кожної з підсистем, що тягне за собою істотне збільшення вартості. Отже, разом з ефективністю необхідно розраховувати і вартісний параметр.

Гарантованість системи, під якою розуміється обов'язковість забезпечення заданого рівня основних експлуатаційних характеристик як системою в цілому, так і її підсистемами і технічними засобами. Вимога гарантованості об'єднує кілька нерівнозначних характеристик як самої системи, так і її підсистем. У самому загальному випадку до основних характеристик можуть бути віднесені надійність, пропускна здатність (швидкодія) і ряд інших.

Уніфікованість системи, під якою розуміється структурна, логічна та інформаційна сумісність, а також можливість нарощувати і удосконалювати систему в процесі експлуатації, покращуючи її характеристики.

Стійкість системи - властивість, що характеризує здатність системи ефективно функціонувати в умовах випадкових і/або навмисних завадових та пошкоджуючих впливів або відновлювати дану здатність протягом заданого часу. Стійкість вважається однією з найважливіших характеристик будь-яких складних систем. Як правило, вона включає в себе дві найважливіші характеристики - живучість (стійкість) і завадозахищеність. При цьому обидві характеристики визначаються в умовах навмисних несанкціонованих дій (пошкоджуючих впливів), спрямованих на порушення функціонування системи охорони периметра. Іншими словами, стійкість повинна визначатися в умовах активного впливу на систему порушника.

Система заходів щодо забезпечення охорони периметра, яка задовольняє сукупності перерахованих вимог, включаючи ефективність і вартість, повинна бути адекватна категорії об'єкта, потенційним загрозам і моделям порушників.

1.2 Вимоги до охоронних систем відеоспостереження швидкого розгортання.

Істотними відмінностями охоронних систем швидкого розгортування, як класу, від традиційних систем охорони периметрів об'єктів є:

- малий час установки елементів системи на місцевості (як правило, не більше 1,5 години для штатного комплекту обладнання);
- можливість швидко змінювати на місцевості розташування датчиків, а значить і конфігурацію контрольованого кордону (району) в залежності від змін обстановки;
- можливість установки на непідготовленою в інженерному відношенні місцевості.

Ці відмінності визначають, що швидкорозгортувані системи призначені для попередження про наближення порушника до об'єкта або про знаходження його в деякому контрольованому районі. Тому такі системи можна використовувати як самостійно, так і спільно з традиційними системами охорони периметрів.

Основні вимоги до охоронних систем відеоспостереження швидкого розгортання визначимо наступним чином[2]:

1. Висока швидкість розгортання системи.

2. Висока гнучкість системи захисту (розмір та форма контрольованої зони)

3. Надійна робота системи у випадку складного рельєфу або щільної забудови

4. Висока адаптивність до несподіваних випадків як підчас розгортання так і підчас штатного режиму роботи.

5. Захист системи від глушіння та саботажу

6. Висока автономність системи

7. Можливість компонування різними типами компонентів, з однаковими інформаційними характеристикам

Сучасні технології та елементна база дозволяють забезпечити зниження габаритно-вагових параметрів елементів системи, вдосконалення алгоритмів обробки сигналів подальші дослідження мають бути спрямовані на розробку нових принципів побудови ОСШР та обґрунтування її структури і параметрів.

1.3 Модель комплексу відеоспостереження швидкого розгортання.

В якості СВШР, яка задовольняє всім висунутим вимогам розглянемо наступну схему комплексу (Рис. 1.2):

1. Для створення периметру охорони та виявлення можливих місць його порушення використовуємо різні набори датчиків швидкого розгортання сигналізаційного принципу дії: обривні, радіохвильові, сейсмоакустичні, інфрачервоні і т.п.
2. Для перевірки події порушення безпеки, фото/відео фіксації факту порушення та спостереження за діями зловмисників використовуємо, як мобільні відеокамери розміщені на БПЛА так і статично закріплені відеокамери.
3. Використання інших БПЛА відеоспостереження та статично закріплених камер в якості ретрансляторів сигналу, для організації каналу зв’язку з центром управління та моніторингу (в разі необхідності).

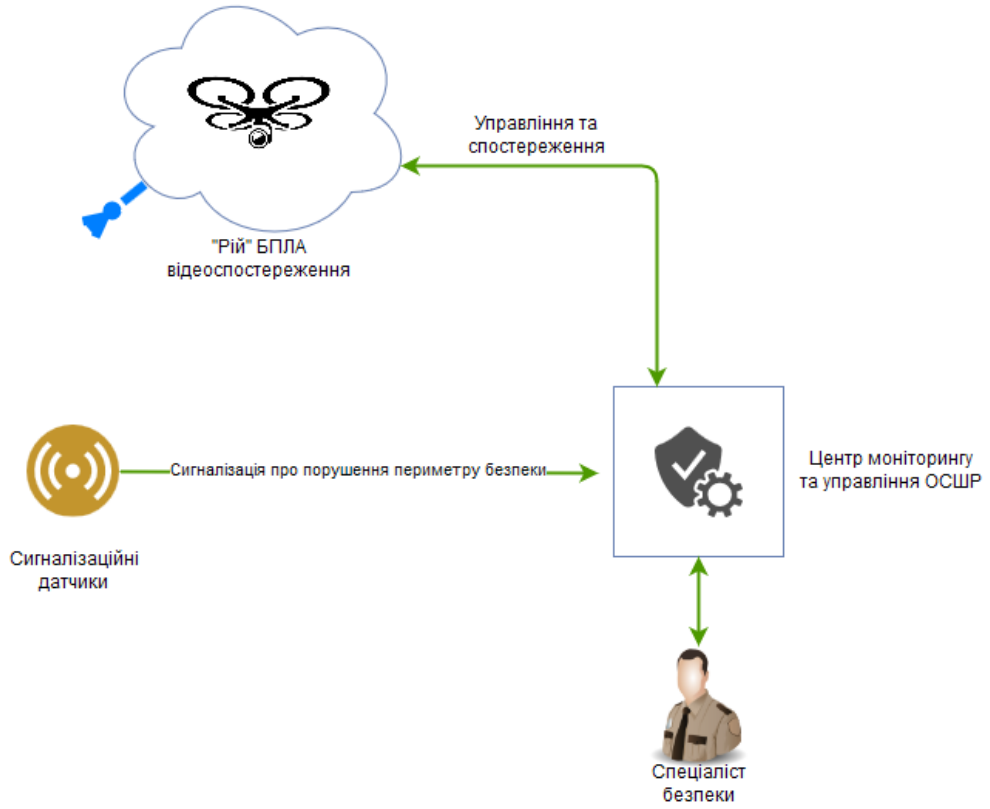


Рис. 1.2 Модель-схема комплексу СВШР

Для подібних систем відеоспостереження можна виділити наступні характерні проблеми та складності, які необхідно подолати при проектуванні:

1. Для передачі відео-фото матеріалів та управління БПЛА потрібні канали зв'язку високої пропускної здатності, які складно та затратно організувати, особливо для супутникового зв'язку.
2. Вразливість каналів зв’язку – будь-які сигнали, які приймаються та відправляються літальним апаратам можна глушити, перехватувати, підміняти.

Висновки по розділу

В запропонованій охоронній системі відеоспостереження швидкого розгортання відеокамери використовуються наступним чином:

1. Поєднання відеокамери з охоронним датчиком.
2. Використання відеокамер для перевірки спрацювання датчику.

Важливим фактором для роботи СВШР є можливість підтримки надійного каналу зв’язку з центром керування та моніторингу з будь-якої точки всередині охоронного периметру, у випадку відсутності прямого радіо-каналу.

Для проектування гнучкої охоронної системи відеоспостереження необхідно вирішити задачу побудови самоорганізаційної сітчатої мережі (mesh-мережа) та забезпечити захист інформації при передачі в ній.

РОЗДІЛ 2
ОГЛЯД ПРОТОКОЛІВ МАРШРУТИЗАЦІЇ ТА МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ В MESH-МЕРЕЖАХ

2.1 Класифікація протоколів маршрутизації

Самоорганізовані мережі являють собою альтернативу інфраструктурних мереж. У такій мережі кожен вузол мережі може виконувати функції маршрутизатора. Можливість кожного вузла покинути мережу або підключитися до неї за власним бажанням призводить до того, що важливим питанням при організації роботи самоорганізованої мережі є вибір протоколу маршрутизації [34]. Розроблені протоколи маршрутизації класифікуються за підходом до оновлення інформації про топології мережі на реактивні, проактивні та гібридні [12,27]. На Рис. 2.1 показана класифікація протоколів маршрутизації.

Реактивний підхід до маршрутизації передбачає побудову маршрутів по мірі їх потреби. При спробі з'єднання з вузлом мережі відбувається повний перебір всіх варіантів і пошук найкращого маршруту до нього відповідно до метрикою маршрутизації. Цей маршрут використовується до тих пір, поки існує зв'язок з адресатом.

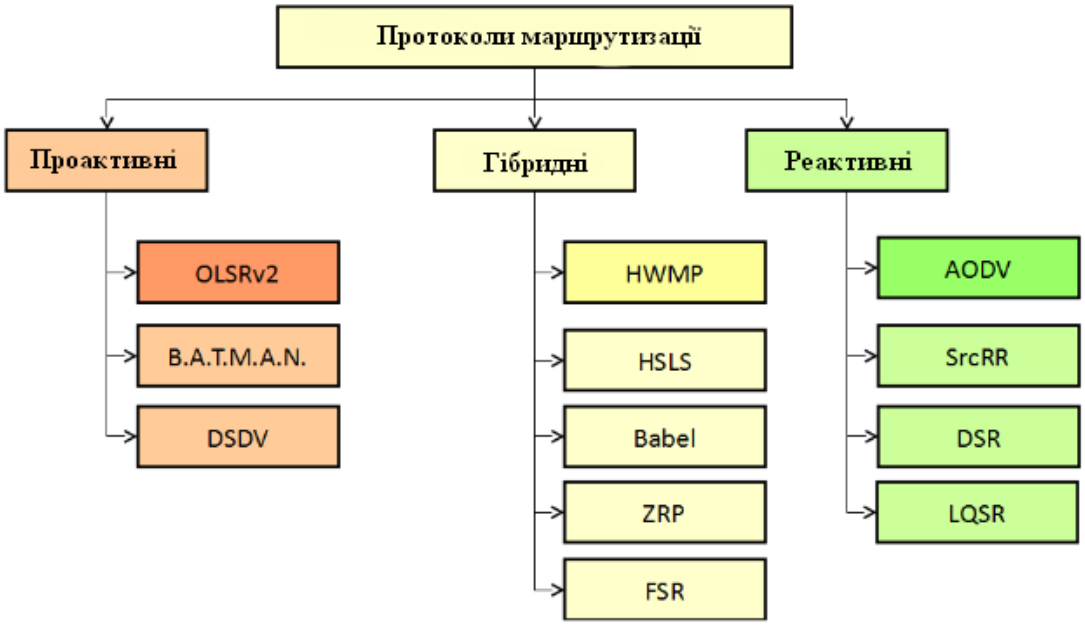


Рис. 2.1 Протоколи маршрутизації в самоорганізованих мережах.

При використанні проактивного підходу топологія мережі має відстежуватися і оновлюватися через певні інтервали часу. Проактивні протоколи оновлюють топологію мережі за допомогою періодичних запитів. Протоколи, що належать до цієї групи, можуть використовувати різну кількість баз даних з інформацією про топологію мережі і різні способи підтримки актуальності цієї інформації. Проактивний підхід спирається на відстеження топології мережі, тому вузли постійно обмінюються повідомленнями, що може привести до підвищення енергоспоживання в порівнянні з реактивним підходом. З іншого боку, вузол мережі, який використовує реактивний підхід, змушений чекати, поки буде проведений перебір варіантів маршруту, що може позначитися на швидкість передачі в мережах зі змінною топологією. Гібридний підхід має на увазі під собою поєднання реактивного і проактивного підходів в рамках однієї мережі.

Вибір найкращого маршруту між вузлами мережі відбувається на підставі метрик: кількості кроків маршрутизації, ETX, ETT, Air Time Link та інших. Метрики можуть враховувати інформацію фізичного, канального і мережевого рівнів моделі OSI.

При обчисленні метрики ETX (Expected Transmission Count) застосовують коефіцієнт доставки кадрів DR (Delivery Rate) прямого df і зворотного dr каналів зв'язку. Для вимірювання DR використовується передача тестових кадрів:

$$ETX = \frac{1}{d_f d_r} \tag{2.1}$$

Метрика ETT (Expected Transmission Time) вдосконалює ETX, додатково враховуючи розмір пробного кадру S і пропускну здатність з'єднання B:

$$ETT = \frac{ETX \cdot S}{B} \tag{2.2}$$

Метрика ALM (Airtime Link Metric) вираховується для пари вузлів та показує непродуктивні витрати при передачі між двома вузлами мережі:

$$c_a = \left(O_{ca} + O_p + \frac{B_t}{r}\right)\frac{1}{1-e_{pt}} \tag{2.3}$$

Вартість з'єднання c_a визначається швидкістю модуляції r та ймовірністю бітової помилки e_{pt} для тестового кадру розміром B_t біт. Непродуктивні витрати, зв'язані з доступністю каналу зв'язку та протоколом, позначені як O_{ca} і O_p відповідно. O_{ca}, O_p і B_t являються константами для кожного типу модуляції, наприклад, для стандарту 802.11a ці значення складають 75 μs, 110 μs і 8224 біта.

2.2 Проактивні протоколи

Протокол OLSR (Optimized Link State Routing) являється проактивним і орієнтований на використання у великих мережах з високою щільністю вузлів. Кожний вузол використовує широкомовне повідомлення HELLO, які передаються через певний проміжок часу до вузлів на відстані одного кроку маршрутизації. Після прийому HELLO вузол одержувач намагається встановити двостороннє з'єднання з вузлом-відправником. Кількість управляючих повідомлень в OLSR знижено завдяки використанню підходу MRP (Multipoint Relays) [23]. В OSLRv2 обмін управляючими повідомленнями в мережі став ще більш ефективним, а сама форма повідомлень була стандартизована та спрощена. OLSR взаємодіє з мережевим рівнем, управляючи таблицями маршрутизації і використовуючи IP адреси для передачі пакетів.

Протокол B.A.T.M.A.N (Better Approach To Mobile Ad hoc Networks) також використовує проактивний підхід [24], в якому всі вузли виконують широкомовну відправку повідомлень OGM (Originator Message). OGM містить адрес ініціатора, адрес одержувача та унікальний послідовний номер. Кожний сусідній вузол змінює адрес отримувача на свій власний та відправляє повідомлення назад ініціатору. Повідомлення OGM не містить в собі ніякої додаткової інформації, наприклад, метрик QoS та таблиць маршрутизації. Протокол B.A.T.M.A.N. має менші непродуктивні витрати в мережах з великою кількістю вузлів, ніж протокол OLSR.

Одним з перших проактивних протоколів був DSDV (Destination-Sequenced Distance Vector), розроблений в 1994 році [25]. Його головною особливістю було додавання поля порядкового номера в керуючі повідомлення, тому що це

дозволило обійти проблему зациклення при розриві з'єднання між вузлами мережі (Loop free), так як тепер кожен вузол знав застаріла чи його інформація про топологію мережі. DSDV виявився неефективний у великих мережах з швидко змінюється топологією, але вплинув на розробку інших протоколів, наприклад, AODV.

2.3 Реактивні протоколи

В реактивному протоколі DSR (Dynamic Source Routing) використовується спеціальний формат заголовку DSR Options Header Format, котрий може бути добавлений до любого пакету і містить маршрут від джерела до вузла призначення [27]. Вузол може виконати процес знаходження маршруту до необхідного вузла (Route Discovery) за допомогою широкомовних повідомлень. Процес підтримки маршруту (Route Maintenance) полягає у відслідковуванні повідомлень канального рівня. Якщо прийнято повідомлення канального рівня або ж залишені без відповіді запити вузла, то процес виявлення повторюється. До недоліків і переваг DSR можна віднести його реактивність, що знижує витрати на передачу керуючих повідомлень, але робить необхідним буферизацію пакетів на час виявлення маршруту. Крім того, спеціальний формат заголовка може привести до того, що при малому обсязі корисного навантаження великий заголовок буде знижувати ефективність роботи мережі.

Подальшим розвитком реактивного підходу став протокол AODV [26]. Замість того, щоб покладатися на передачу об'ємних заголовків, в AODV були заново введені таблиці маршрутизації, які накопичували всю інформацію про топології мережі у міру отримання повідомлень від інших вузлів. Для уникнення зациклення були введені два порядкових номери: для джерела і для адресата, що дозволило відстежувати новизну інформації про топології при використанні маршруту від адресата до джерела. Використання AODV рекомендовано для мереж від 10 до 1000 мобільних вузлів. Головною метою його розробки було зниження витрат на передачу керуючих повідомлень і поліпшення масштабованості та продуктивності роботи мережі.

Іншим протоколом, заснованому на DSR, став реактивний SrcRR. Його головною відмінністю від DSR стало використання метрики ETX, яка вимірювалася за допомогою періодичних широкомовних розсилок до сусідніх вузлів, а для всього маршруту використовувалася сумарна ETX його частин. Крім того, SrcRR не залежав від мережевого рівня і міг використовувати для пошуку шляху MAC-адреси.

Компанія Microsoft розробила і запатентувала протокол LQSR (Link Quality Source Routing), який теж заснований на DSR [31]. Він реалізований між канальним і мережевим рівнем за допомогою віртуального мережевого адаптера, що дозволяє працювати відразу з декількома фізичними сполуками. Тема LQSR розташований між Ethernet-заголовком і корисним навантаженням кадру. Кожен вузол, як і в SrcRR, вимірює метрику QoS до сусідніх вузлів, поширює цю інформацію по мережі, і вона враховується при виборі кращого шляху до адресата. Керуючись правилом, що найкоротший шлях не означає найкращий, LQSR дозволяє використовувати три метрики QoS: ETX, RTT і PktPair.

2.4 Гібридні протоколи

Гібридний підхід дає можливість використання реактивного і проактивного підходів в рамках однієї мережі. Він застосовується в стандарті 802.11s для забезпечення підтримки WMN на канальному рівні [15]. У попередніх стандартах сімейства 802.11 не було можливості отримання метрик QoS канального рівня. Для того, щоб метрика QoS точніші параметри, її слід отримувати на більш низькому рівні мережевої моделі OSI.

Як протокол за замовчуванням в стандарті 802.11s рекомендований гібридний HWMP (Hybrid Wireless Mesh Protocol), а опціональним протоколом може виступати OLSR. Реактивний підхід реалізований на основі AODV (Ad hoc On-demand Distance Vector). В цьому випадку вузол шукає найкращий маршрут у міру потреби, беручи до уваги метрики QoS. При використанні проактивного підходу в WMN призначається кореневої вузол (Root), який виробляє опитування вузлів з певним інтервалом, таким чином, оновлюючи карту мережі.

Підключившись вузол, може зв'язатися з кореневим вузлом і отримати інформацію про маршрутах до всіх вузлів мережі. Обидва підходи можуть бути використані як окремо, так і одночасно в одній мережі.

Гібридний підхід був використаний не тільки в HWMP, але і в більш ранніх протоколах, наприклад, HSLS (Hazy Sighted Link State) роутінг протокол. Преслідуючи мету зниження непродуктивних витрат, HLSL контролює інтервал оновлення інформації про топології мережі, щоб скоротити кількість керуючих повідомлень [33]. Якщо маршрут застарів, то HLSL починає працювати в реактивному режимі. Відсутність актуальної інформації про топології мережі являє собою головний недолік даного протоколу.

Іншим гібридним протоколом для WMN є Babel. Грунтуючись на ідеях DSDV, AODV і протоколу Cisco EIGRP (Enhanced Interior Gateway Protocol), Babel дотримується проактивного підходу і спрямований на роботу в мережах з мобільними вузлами [28]. Він дозволяє реалізувати різні метрики QoS, хоча за замовчуванням використовує ETX. Реактивний режим застосовується в Babel, якщо жоден маршрут від вузла не є придатним для надійної передачі пакетів.

У гібридному протоколі ZRP (Zone Routing Protocol) вузол застосовує проактивний пошук маршрутів в рамках певної ділянки мережі і реактивний за його межами [30].

Протокол FSR (Fish-eye State Routing Protocol) характеризується тим, що точність інформації про топології мережі зменшується в міру віддалення від вузла [31].

2.5 Дослідження принципу роботи протоколу маршрутизації HWMP (стандарт 802.11s)

Стандарт 802.11s дозволяє Wi-Fi-пристроям самоорганізовуватись та автоматично налаштовувати топологію мережі. Wi-Fi-пристрої mesh-мережі називаються сітчатими станціями (STA – Station Networking). STA, які находяться далеко один від одного, можуть зв’язуватись один з одним

використовуючи бездротову маршрутизацію, коли пакети даних передаються через проміжні вузли.

Гібридний безпровідний протокол mesh-мережі (HWMP) є базовим протоколом для маршрутизації в стандарті 802.11s. Він реалізований на канальному рівні моделі OSI [12].

Сітка STA може автоматично і ефективно формувати бездротове з'єднання. Найкращий маршрут може бути знайдений за допомогою HWMP, який оснований на алгоритмі найкоротшого шляху. Для знаходження найкоротшого шляху використовуються алгоритми Беллмана-Форда або Дейкстри [13]. Для вибору оптимальних маршрутів в мережі використовують різні критерії (метрики). Метрики можуть включати в себе наступну інформацію:

- довжина шляху
- надійність
- затримку
- пропускну здатність
- загрузку
- вартість передачі трафіку

Гібридний протокол маршрутизації HWMP використовує стандартний набір службових пакетів, правил їх створення та обробки, подібно до потоколу дистанціонно-векторної маршрутизації по запиту (Ad Hoc On Demand Distance Vector, AODV) [14]. HWMP адаптований для роботи з адресами MAC-рівня та метриками шляху. Гібридним він названий тому, що об'єднує в собі два режими побудови шляху, котрі можуть бути використані як окремо один від одного, так і одночасно в одній мережі:

реактивний режим – побудова маршрутних таблиць у вузлах mesh-мережі безпосередньо перед передачею даних (на основі запиту).

проактивний режим – регулярна процедура обновлення інформації в маршрутних таблицях вузлів всієї мережі. Процедуру ініціює кореневий вузол, в результаті будується граф (дерево) шляхів з вершиною в кореневому вузлі.

Існує чотири типи кадрів в HWMP, які безпосередньо беруть участь в процесі знаходження шляху:

1. Path Request (PREQ) – шлях запиту
2. Path Reply (PREP) – шлях відповіді
3. Path Error (PERR) – помилка шляху
4. Root Announcement (Rann) – кореневе об’явлення.

На Рис. 2.2 зображені формати кадрів в HWMP [11].

Кожний з цих кадрів має наступні поля: довжина кадру (Length) та флаг (Flags), який визначає формат передачі (01 – роль порталу, 10 – групова передача, 11 – індивідуальна передача). Поле Hope Count в кадрах PREQ, PREP, RANN визначає кількість проміжних вузлів при передачі від вузла-відправника до вузла призначення. Якщо вузол призначення не знайдений, то в поле Element ID встановлюється значення Error. Поле PREQ ID містить унікальний ідентифікатор кадру PREQ. Поле Originator Address містить MAC-адресу відправника. Поле порядкового номеру ініціатора (Originator Sequence Number) кодується як ціле без знаку і містить порядковий номер, специфічний для відправника. Поле Originator External Address визначає MAC-адресу проксуємого об’єкту у випадку, якщо PREQ-кадр був згенерований за границею mesh-мережі. Поле Lifetime визначає час на протязі якого кадр вважається дійсним. Target count дає кількість направлень, що містяться в цьому PREQ. Поле Target Address представляється, як 48-бітна MAC адреса вузла призначення.

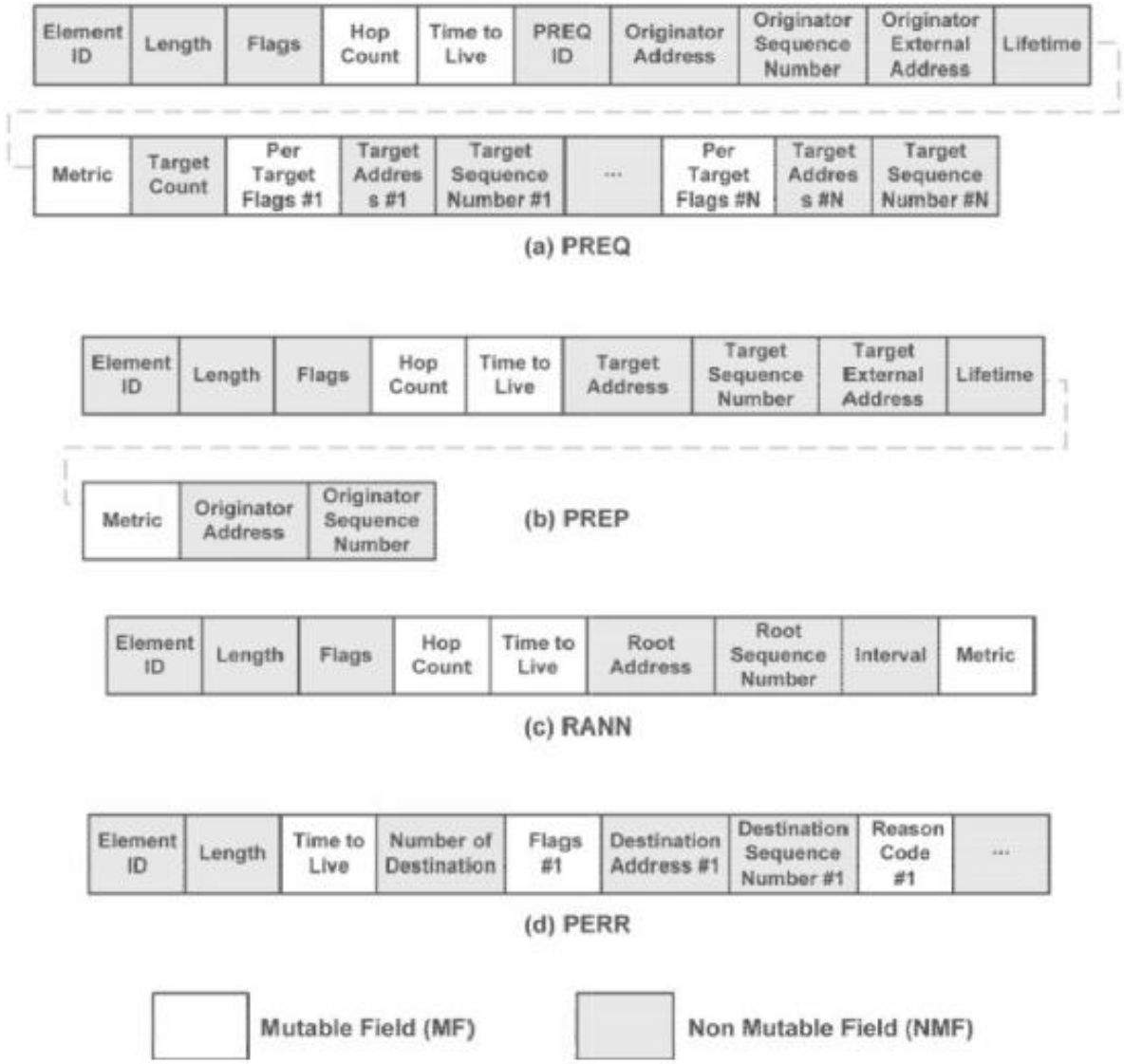


Рис. 2.2. Формат кадрів в HWMP

В кадрах HWMP є змінні (MF – Mutable Field) та незмінні (NMF – Non Mutable Field) поля [13]. MF містить інформацію, котра буде оновлюватись по мірі розповсюдження кадрів в мережі. NMF містить інформацію, яка не може бути змінена в проміжних вузлах STA.

В реактивному режимі HWMP вузол відправляє широкомовний PREQ-кадр. Шляхи вибираються на основі метрики, для розповсюдження інформації слугує спеціальне поле в службових пакетах запиту шляху. Цей пакет розповсюджується через сусідні вузли по всій мережі, поки не досягне вузла призначення (Рис. 2.3) [13,21]. По мірі проходження кадру від вузла до вузла модифікується поле метрики шляху. В результаті формується повна метрика

шляху «одержувач-відправник». Вузел-адресат може відправляти ініціатору пакет підтвердження PREP, який містить результуюче значення метрики шляху «відправник-одержувач», в цьому випаду з’єднання являється зареєстрованим (Рис. 2.4). Прийнявши його, вузел-ініціатор отримує інформацію про встановлений шлях. У реактивному режимі пакети підтвердження PREP можуть відправляти не тільки вузел призначення, але й всі проміжні вузли, успішно прийнявши пакет запиту PREQ (якщо в пакеті встановлені відповідні мітки).

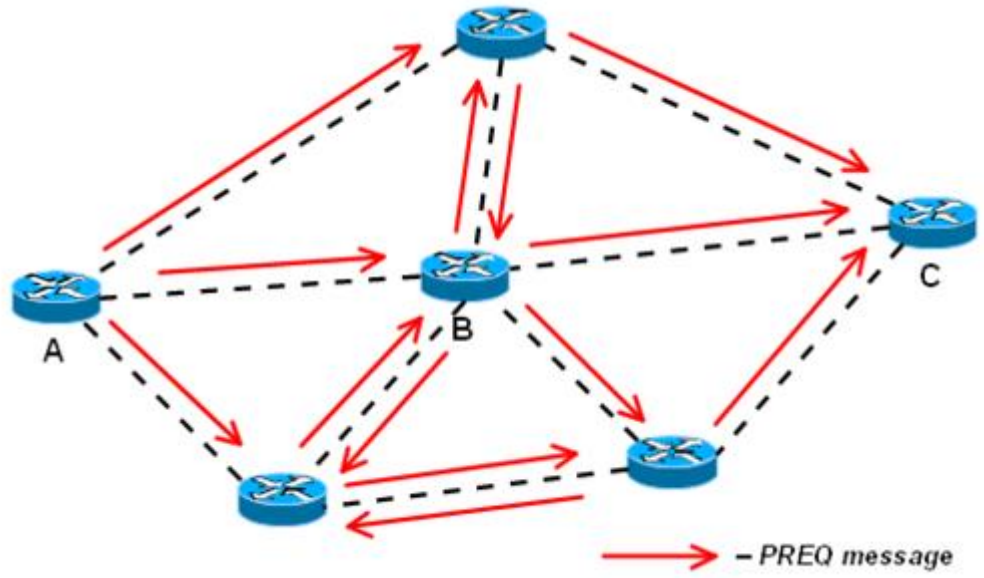


Рис. 2.3 Вузел А визначає шлях до вузла С

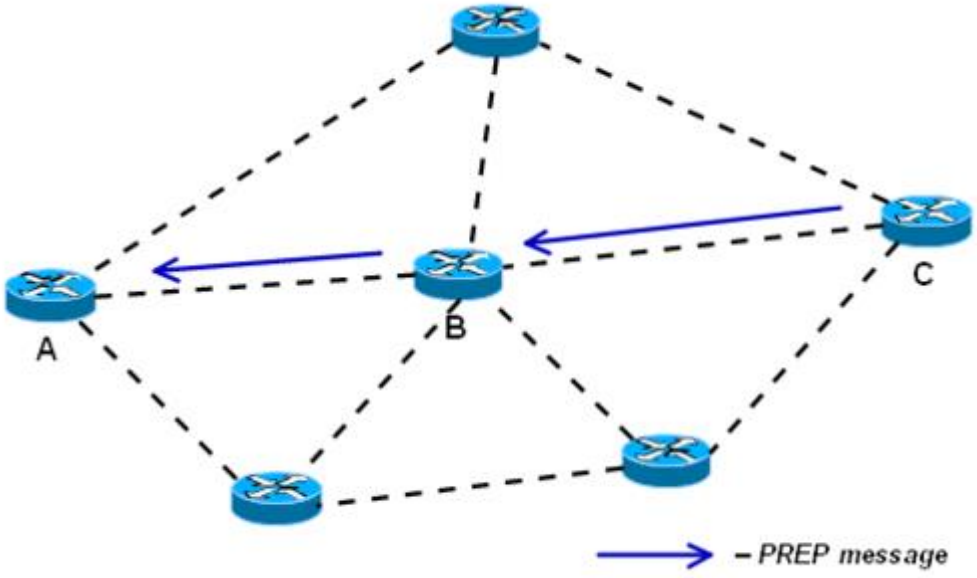


Рис. 2.4 Вузел С відправляє відповідь вузлу А.

Всі вузли mesh-мережі зберігають інформацію про кожний вузол мережі, оновлюючи її на основі отриманих службових пакетів. Така інформація в даних пакетах передається в полях «адрес відправника», «метрика шляху», «порядковий номер запиту» (Originator's DSN (OSN)) [15].

Крім полів метрики шляху по мірі проходження від вузла до вузла в пакеті може змінюватись значення поля «час життя» (Time to Live, TTL). Якщо цей параметр використовується, то він зменшується в кожному вузлі, через який проходить пакет. В маршруті слідування пакетів можуть бути замкнуті маршрути (цикли). Щоб уникнути таких циклів, використовується порядковий номер запиту. Цей параметр слугує номером при розсиланні пакетів пошуку шляху. Кожний mesh-пристрій має свій власний DSN. Перед початком процедури пошуку шляху DSN ініціатора збільшується на 1 та записується в поле Originator's DSN пакета запиту PREQ [12].

Вузол, отримавши пакет PREQ, порівнює значення OSN з раніше збереженими значеннями для цього ж відправника. Пристрій приймає, оброблює та ретранслює пакет PREQ, тільки якщо поточний OSN в пакеті більше раніше збереженого або вони рівні, але метрика шляху раніше отриманого пакету гірше ніж у знову отриманого (тобто повторного прийому і ретрансляції одного і того ж самого пакету не може бути). В реактивному режимі пакети підтвердження PREP можуть відправляти не лише вузол призначення, але і всі проміжні вузли, які успішно прийняли пакет запиту PREQ (якщо в пакеті встановлені відповідні мітки).

В проактивному режимі назначається корневий вузол (або вузли). Цей вузол періодично розсилає пакети PREQ, які розповсюджуються по всій мережі (Рис. 2.5). Всі вузли мережі, прийнявші проактивний PREQ, зберігають адресу вузла-відправника і широкомовно транслюють PREQ зі зміненими полями (поля метрики та TTL) та відправляють PREP корневому вузлу, якщо встановлений відповідний флаг в PREQ (Рис. 2.6). Крім описаних методів вибору шляху на основі пакетів PREQ та PREP стандарт передбачає процедуру на основі пакетів

оповіщення про корневий вузол Rann. Цей метод принципово не відрізняється від вже розглянутого.

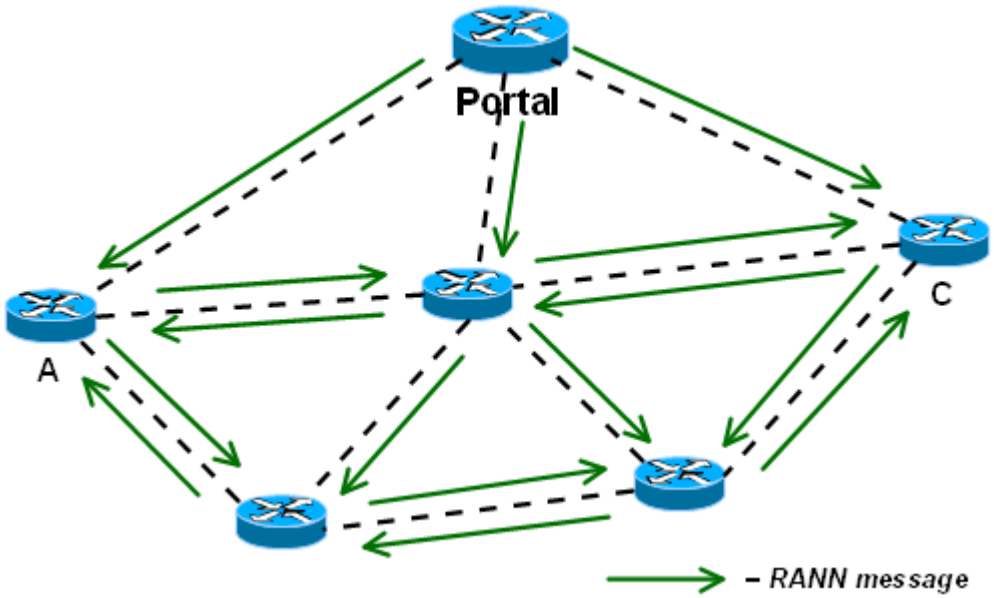


Рис. 2.5 Кореневий вузол анонсує себе відправляючи RANN/PREQ-пакети

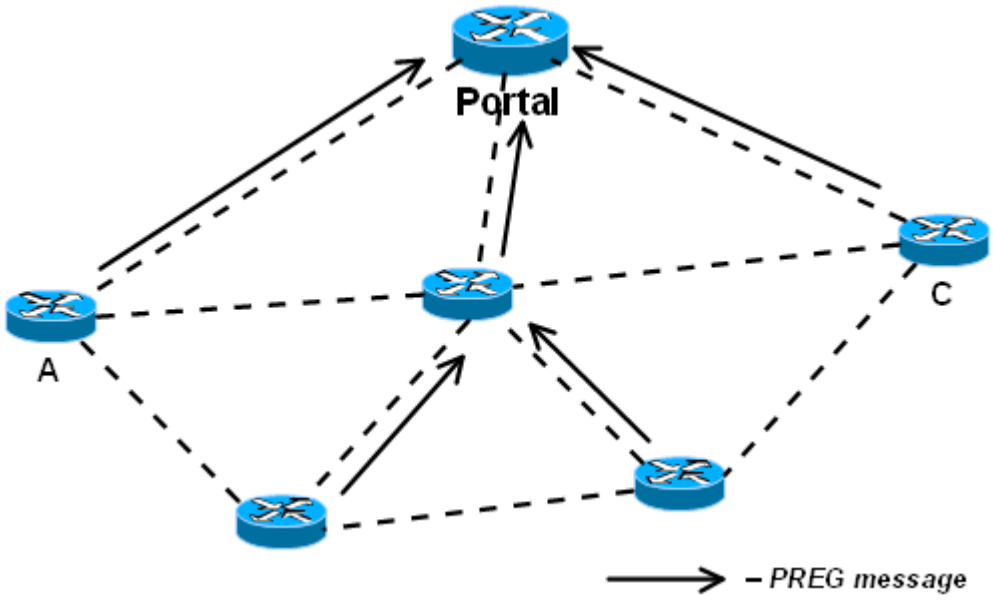


Рис. 2.6 Внутрішні вузли відповідають за допомогою PREP-пакетів

2.6 Вразливості протоколу маршрутизації HWMP

У той час як принципи роботи HWMP визначені в стандарті, функції безпеки не описуються для цього протоколу [11]. У міру поширення кадрів по мережі деякі частини кадрів можуть бути змінені в проміжних вузлах. Найбільш часто змінені частини повідомлення маршрутизації включають лічильник

переходів і метрику шляху запитуваної вузла. З точки зору безпеки проміжні вузли мережі не є довіреними вузлами. Для забезпечення безпеки маршрутизації повідомлень, для двох типів частин кадрів, змінюваних і незмінних полів, потрібні різні вимоги захисту. Змінні поля повинні оновлюватися відповідно до правил маршрутизації в міру просування кадрів в мережі. Кожен вузол вимагає криптографічного захисту для виявлення незаконно-зміненої інформації в повідомленні. Безпека незмінних полів і цілісності даних повинна здійснюватися автентифікація полів при передачі кадрів між вузлами джерела та призначення.

Існують наступні типи атак на mesh-мережі:

1. Переповнення – зловмисник може транслювати PREQ-кадри безперервно, що викличе переповнення мережі [16]. PREQ-кадри безперервно поширюються по всьому WMN. Атакуючий може використовувати адресу одержувача, якого не існує в WMN, через що PREQ-кадри будуть поширюватися по всій мережі. Зловмисник може також безперервно транслювати Rann-кадри. Це безперервне мовлення змусить інші вузли мережі відповідати на Rann-запити. Якщо шляху відомі, зловмисник може відправити підроблені PERR-кадри [17]. Це помилкове повідомлення про помилку призведе до видалення інформації про маршрут цільового вузла в таблиці маршрутизації. Вузол мережі буде змушений запитувати нові шляхи через підроблених PERR-кадрів.
2. Диверсія – зловмисник може збільшити порядковий номер запиту в PREQ-кадрі, тим самим обдуривши інші сітки STA. В цьому випадку таблиця маршрутизації для кожного вузла буде оновлюватися на основі припущення, що PREQ-кадр містить нову інформацію. Зловмисник також може знизити значення метрики, щоб шлях став краще, коли насправді існують найбільш оптимальні шляхи.
3. Атаки типу Wormhole і Blackhole – в разі Wormhole-атаки зловмисник може стежити за переданими пакетами даних [17]. У Blackhole-атаці передані пакети даних будуть видалені зловмисником під час пересилки [18]. У цих атаках маршрут повинен проходити через вузол зловмисника.

4. Імперсонація – є чотири поля адреси в рамках HWMP: передавач, приймач, відправник і мета [19]. Зловмисник може видавати себе за іншу сітку STA, змінюючи адреси в цих областях. Імперсонація може бути використана для створення циклу, змінивши адреси передавача і приймача. Також атакуючий, видаючи себе іншим вузлом, може ініціювати PREQ-запит. Ця імперсонація виконується шляхом маніпулювання адресом відправника та цілі.
5. Атаки типу Replay (повторення) – зловмисник може перехопити на декількох сеансах зв'язку передані пакети. Відправляючи перехоплені кадри PREQ і підміняючи MAC-адресу жертви, зловмисник переконає вузол призначення в тому, що вузол жертви знову намагається зв'язатися з вузлом призначення. Вузол призначення відповідає з PREP для атакуючого вузла. На даний момент зловмисник починає здійснювати зв'язок з вузлом призначення, а вузол-адресат вважає, що зловмисник є першоджерелом.
6. Підслуховування – HWMP-кадри містять інформацію про маршрутизацію. Інформація про маршрутизацію може бути отримана шляхом прослуховування обміну кадрами в HWMP. У деяких випадках число вузлів мережі в WMN має зберігатися в секреті. Якщо WMN використовується у військовій області, противник може аналізувати військовий потенціал, грунтуючись на кількості вузлів. Крім того, інформація про шляхи маршрутизації може бути використана для аналізу положення і відстані інших вузлів мережі [20]

Атаки на протокол маршрутизації можна розділити на зовнішні і внутрішні. Зовнішні атаки відносяться до нападів зловмисника, який не має доступу до мережі. Внутрішні атаки відносяться до нападів вузла, який пройшов перевірку автентичності в мережі.

Внутрішні зловмисники є вузлами WMN. Внутрішні атакуючі мають всі необхідні ключі безпеки. У цих атаках можуть бути застосовані всі типи зовнішніх атак. Виявити і запобігти внутрішнім атакам набагато складніше, ніж зовнішні. Реальною проблемою в WMN є можливість внутрішньої атаки.

Внутрішній зловмисник може також претендувати на роль кореня в мережі. Коли сітка STA намагається посилати пакети, він буде ініціювати процес виявлення шляху до мети. Якщо немає шляху в маршрутних таблицях відправника, вузлик буде передавати пакети корені. Таким чином, можна буде стежити за першими декількома пакетами даних.

Існують наступні розроблені протоколи безпеки кадрів для HWMP:

- Протокол цілісності широкомовлення (BIP)
- Блочний шифр повідомлень коду автентифікації (CCMP)
- Безпечний HWMP (SHWMP)
- Ідентифікація на основі шифрування HWMP (IBC-HWMP)
- Еліптичні криві цифрового підпису HWMP (ECDSA-HWMP)

Сторожеві HWMP (Watchdog HWMP)

2.7 Протокол безпеки CCMP

Протокол CCMP шифрує тіло кадру та код цілісності повідомлення (MIC) [11]. Заголовок HDR в CCMP містить інформацію, необхідну для дешифрування та перевірки цілісності , наприклад, номер пакету (Packet Number -- PN), вектор ініціалізації (Initialization Vector – IV) та ідентифікатор ключа. MAC HDR містить MAC-адресу відправника та приймача.

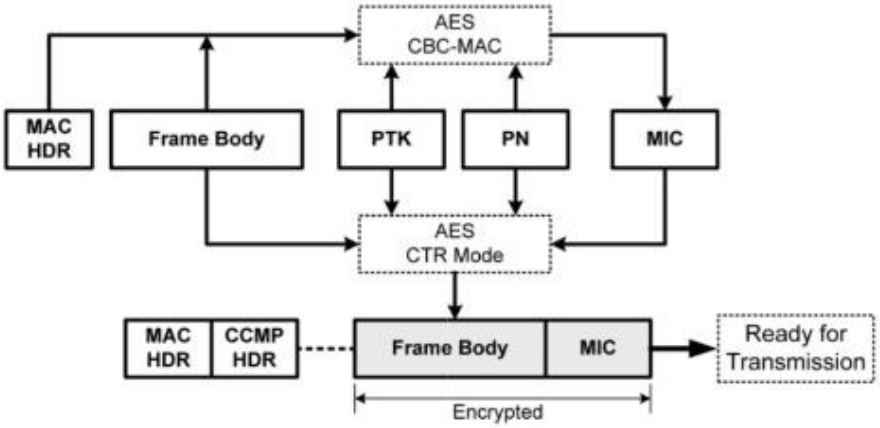


Рис.2.7 Принцип роботи CCMP

2.8 Протокол безпеки BIP

Протокол безпеки BIP вводить цілісність IGTK (Integrity Group Temporal Key) та номер пакету цілісності (Integrity Packet Number – IPN), які використовуються винятково в BIP. Використання IGTK та IPN подібне до використання PTK (Pairwaise Transient Key) та PN в CCMP [22]. Елемент цілісності коду повідомлення (MIC Integrity Element – MMIE) містить інформацію, необхідну для перевірки цілісності, включаючи IPN та ідентифікатор ключа.

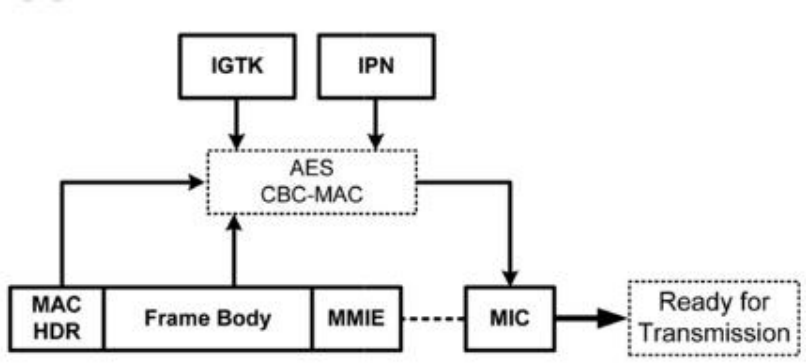


Рис. 2.8 Принцип роботи BIP

Протокол BIP не забезпечує шифрування та призначений для широкомовних кадрів, в той час як CCMP призначений для однонаправлених кадрів.

2.9 Протокол безпеки SHWMP

Безпечний HWMP (Secure HWMP, SHWMP) забезпечує захист кадрів від зовнішніх атак. Для створення коду цілісності MIC змінних полів використовується алгоритм Merkle [16]. Код цілісності шифрується за допомогою PTK/GTK для забезпечення автентифікації та цілісності змінних полів. Незмінні поля також шифруються за допомогою PTK/GTK, які забезпечують конфіденційність кадрів. Для кадрів PREQ лічильник кількості вузлів, час життя (TTL), метрика та флаг призначення являються незмінними елементами. Кількість переходів, TTL та метрика є незмінними елементами для кадрів PREP та Rann. Безпечний HWMP не пропонує служби безпкеи для PERR-кадрів.

Протокол SHWMP концентрується на захисту кадрів від зовнішніх атак. GTK забезпечує захист тільки між пірінговими з’єднаннями. Недоліком протоколу SHWMP є те, що він не забезпечує захист від внутрішніх атак, а також не забезпечує надійну автентифікацію та перевірку цілісності кадрів [17]. Цей протокол також не має якихось схем безпеки для PERR-кадрів. Протокол не забезпечує перевірку справжності та цілісності незмінних полів в кадрах при передачі через проміжні вузли від вузла-джерела до вузла-призначення. Крім того, в SHWMP не існує захисту від атак типу Replay.

2.10 Протоколи безпеки IBC-HWMP та ECDSA-HWMP

В протоколі IBC-HWMP для ідентифікації вузла STA використовують MAC-адресу. Закриті ключі використовуються для шифрування змінних полів кадрів PREQ та PREP. Відкриті ключі розповсюджуються для перевірки підписів. Схема підпису та ECDSA використовуються для створення підписів протоколів IBC-HWMP та ECDSA-HWMP. Для реалізації ECDSA-HWMP в мережі повинен передаватися цифровий сертифікат, що потребує третього довіреного центру сертифікації. Доцільність впровадження центру сертифікації в бездротовій сітчатій мережі викликає сумніви.

Протокол IBC-HWMP не забезпечує захист Rann та PERR-кадрів, захист автентифікації та цілісності незмінних полів при передачі кадрів, захист від атак типу Replay.

2.11 Watchdog HWMP

Watchdog HWMP може знаходити незаконну зміну полів кадрів. При передачі PREQ-кадра вузол-відправник широкомовно транслює цей кадр всі сусіднім вузлам. Сусідні вузли, отримавшки PREQ-кадр, також розпочинають широкомовно транслювати цей кадр. При досягненні кадром PREQ вузла призначення, цей вузол приймає кадри PREQ від сусідніх вузлів та порівнює їх з оригіналом. Це значить, що всі вузли повинні зберігати інформацію про всі PREQ-кадри, що передаються і порівнювати їх зі всіма PREQ-кадрами сусідніх вузлів. По мірі розростання мережі та збільшення кількості вузлів це приводить

до погіршення пропускної здатності, що в значній мірі відображається на ефективності протоколу Watchdog HWMP.

Висновки по розділу

В рамках розглянутої моделі для організації надійного бездротового каналу зв’язку високої пропускної здатності для забезпечення передачі відеоматеріалів від відеокамер важливим фактором являється вибір протоколу маршрутизації сітчатої мережі.

Існує три типи протоколів маршрутизації в самоорганізаційних мережах: проактивні, реактивні та гібридні. Найбільший інтерес для розгляду представляють гібридні протоколи, серед яких найбільш розвинутим є Hybrid Wireless Mesh Protocol (HWMP).

Оскільки в стандарті протоколу HWMP описаний лише принцип роботи, а функціям безпеки та засобам захисту, не приділено необхідної уваги, то існує декілька розробок та протоколів, які забезпечують безпеку протоколу HWMP , проте і вони не забезпечують повноцінність захисту інформації при її передачі.

РОЗДІЛ 3
ВИКОРИСТАННЯ ГЕОМЕТРИЧНИХ АЛГОРИТМІВ ДЛЯ ПОКРАЩЕННЯ ПЕРЕДАЧІ ДАННИХ В MESH-МЕРЕЖАХ

3.1 Загальний опис алгоритмічної структури.

При представленні сітчатої мережі передачі даних у вигляді графу, вузлами будуть виступати передатчики\приймачі(=БПЛА=відеокамери), ребрами графу будуть виступати активні канали передачі даних. Довжиною ребер, в цьому випадку, актуально вибрати метрику на основі не абсолютних координат(GPS), а відносного позиціонування. Позиціонування можна здійснити різними методами, або їх комбінацією – AOA (кут прибуття), TDOA (різниця в часі прибуття), RSSI (показник сили отриманого сигналу). Тут і далі під передачею розташуванням будемо мати на увазі данні на основі яких, можливо здійснити позиціонування заданого вузла відносно інших вузлів графу.

Коли початкова фаза побудови сітчастої мережі буде закінчена, проблеми передачі даних (проблеми мережі), можна трансформувати в геометричні проблеми графів та вирішити за допомогою геометричних алгоритмів.

В рамках запропонованої схеми по усуненню завад зв’язку розглядаються наступні три геометричні алгоритми: Алгоритм вимітання площини (Plane sweep algorithm), алгоритм побудови діаграми Вороного та алгоритм триангуляції Делоне. Алгоритм вимітання площини використовується для перевірки перетину лінії зв’язків між вузлами. Діаграма Вороного використовується для пошуку сусідніх вузлів. Триангуляція Делоне використовується для зменшення довжини зв’язків між вузлами. В самому кінці ми також застосовуємо стандартне відхилення для обрізання занадто довгих зв’язків та покращення пропускної здатності мережі.

Запропонована схема по усуненню завад зв’язку зображена на Рисунку 3.1

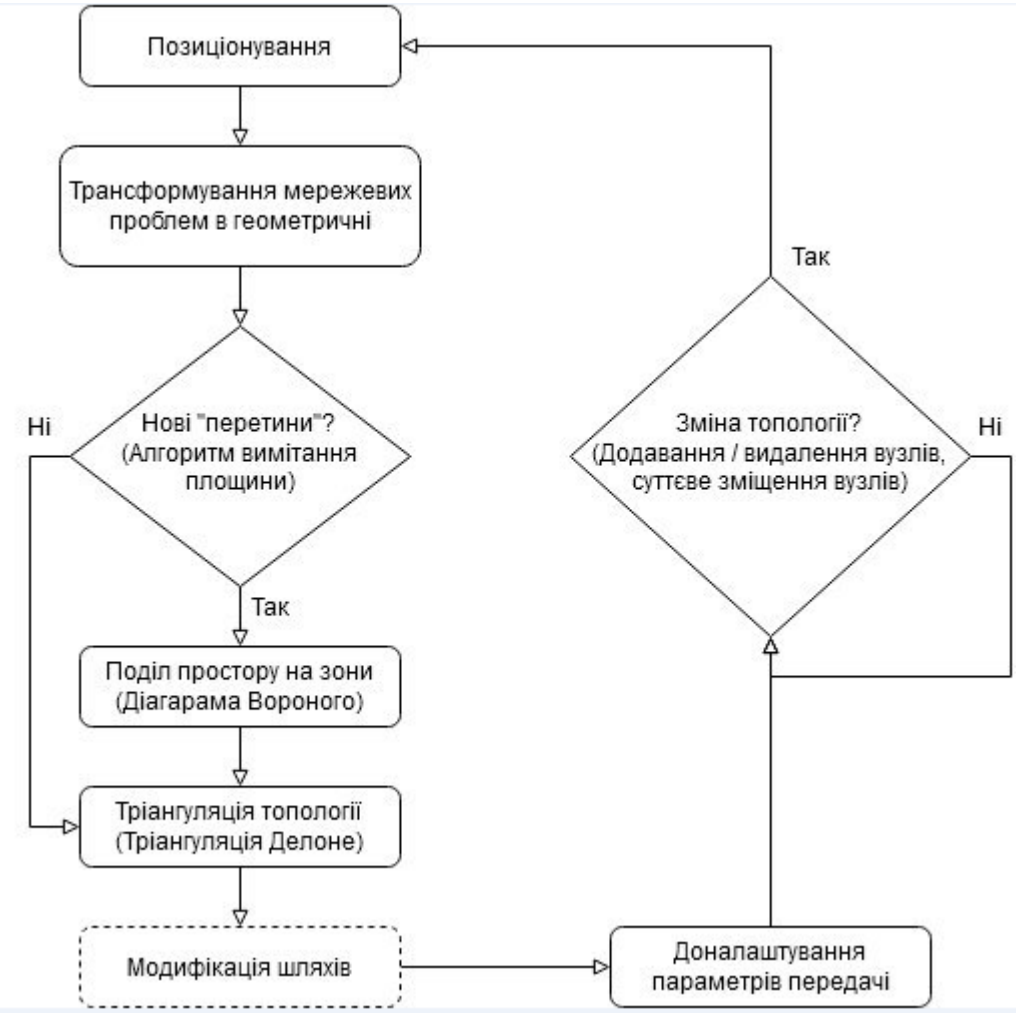


Рис. 3.1. Запропонована алгоритмічна структура для покращенню якості зв'язку.

3.2 Алгоритм вимітання площини

Алгоритм вимітання площини використовується для перевірки перетину ліній на геометричному графі. Нумеруємо всі лінії зв’язку та скануємо всі лінії на графі зверху вниз. Коли знаходимо кінцеву точку лінії (точку події) ми додаємо цю точку в чергу бінарного дерева подій (Рисунок 3.2). Черга використовується для перетину ліній. Точка події не буде видалена з черги, поки лінія цієї точки події не пройде сканування.

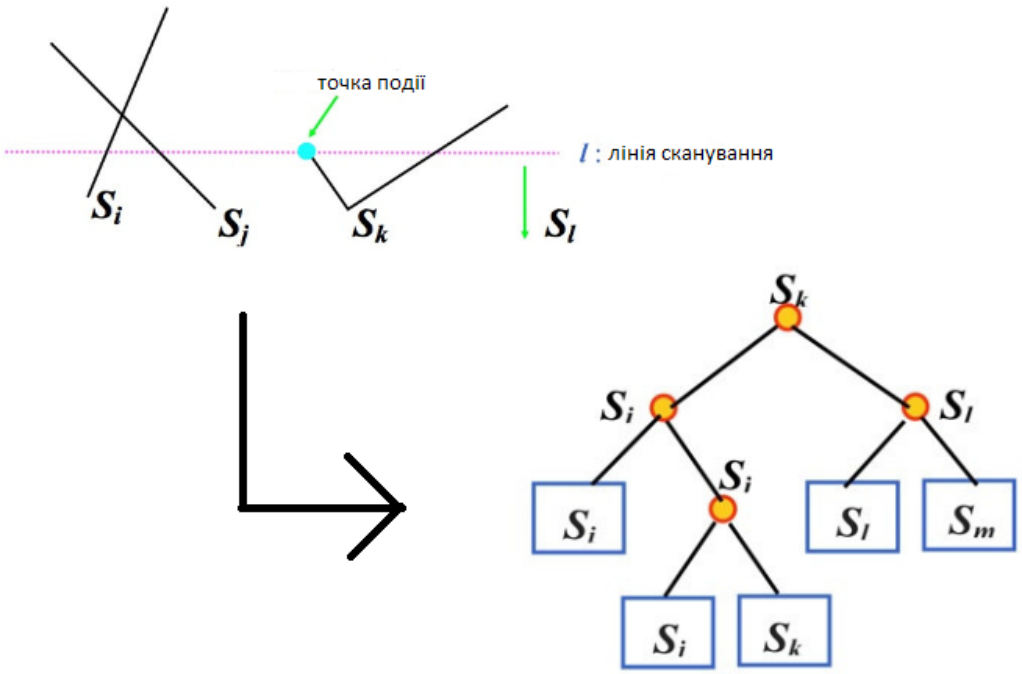


Рис. 3.2. Пошук точок подій та побудова відповідного бінарного дерева.

Алгоритм вимітання площини має складність O((n+I) log n), де n – кількість точок подій; I – кількість точок перетину ліній. Для порівняння складність алгоритму повного перебору складає $O(n^2)$.

3.3 Діаграма Вороного

Діаграма Вороного для графу має властивість, згідно якої найближчим вузлом для будь-якої точки x в області $V(P_i)$ є сам вузел P_i. Ребра на границі областей $V(P_i)$ та $V(P_j)$ являють собою рівновіддалену множину точок від вузлів P_i та P_j. Вершини ребер – точки рівновіддалені від 3х відповідних вузлів графу. Складність побудови діаграми Вороного складає O(n log n), де n – кількість вузлів графу.

Математично однозначно визначення діаграми Вороного можна дати наступним чином: P = {P1, P2,...,Pn} – множина вузлів графу, тоді V(P)={V(P1),V(P2),.....,V(Pn)} буде називатися діаграмою Вороного, якщо кожний V(Pi) визначається наступним чином:

$$V(Pi) = \{x: (P_i - x) < (P_j - x), \forall j \neq i\} \quad (3.1)$$

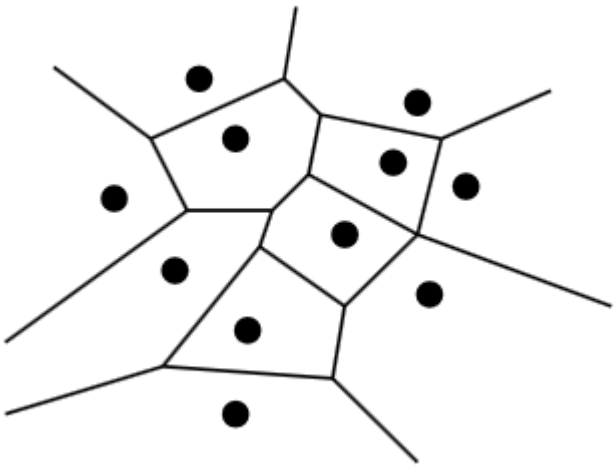


Рис. 3.3. Приклад розділення вузлів за допомогою діаграми Вороного.

3.4 Тріангуляція Делоне

Цей алгоритм буде застосовано до нашої проблеми у двох випадках: при виконанні трансформації топології графу та у випадку реконструкції топології після додавання, видалення вузлів або істотній зміні топології за рахунок переміщення. Тріангуляція Делоне використовує алгоритм легалізації тріангуляцій [35] для модифікації полігонів.

На Рисунку 3.4 зображено два випадка тріангульованих полігонів. Ребро P_iP_j називає «нелегальним» якщо виконується співвідношення $(\alpha_2 + \alpha_5) > (\beta_1 + \beta_6)$; відповідно ребро P_lP_k вважається легальним. Легалізацією тріангуляцій називається алгоритм по зміні всіх нелегальних ребер, на легальні у відповідних полігонах.

Складність тріангуляції Делоне складає O(n log n), де n – кількість вузлів графу.

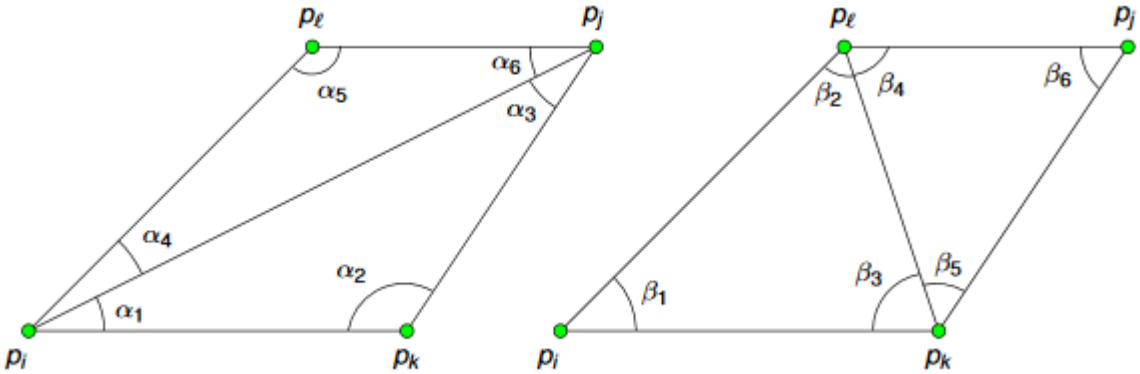


Рис 3.4 Тріангуляція полігонів

3.5 Добавлення та видалення вузлів графу

Коли новий вузол добавляється в граф, він перш за все широкомовно розповсюджує інформацію про своє розташування до сусідніх вузлів. Інформацію про появу нового вузла в системі запускає перепроходження алгоритму вимітання площини та перерахунок тріангуляцій для отримання актуальної топології. Можливо два випадки при додаванню вузла:

a) Новий вузол в середині існуючої тріангуляції.
b) Новий вузол на межі двох суміжних тріангуляцій (Рис 3.5).

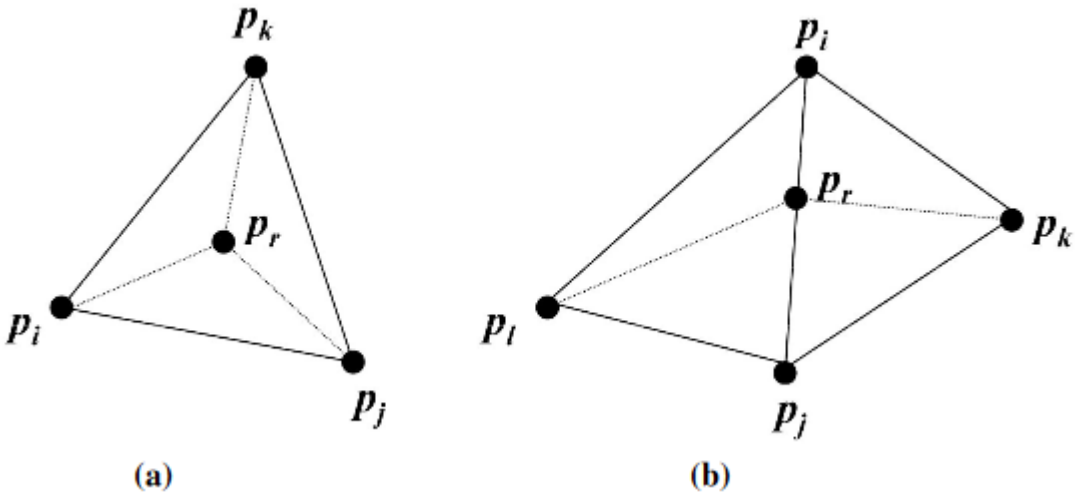


Рис 3.5 Варіанти добавлення нового вузла в граф.

На Рисунку 3.6 зображена перебудова топології після добавлення вузла p_r. Процес видалення вузла та відповідна реконструкція топології проходить аналогічний чином до процесу добавлення нового вузла. Сигналом для запуску процесу перебудови топології є ситуація, коли сусідні вузли не отримують оновленої інформації від вузла на протязі певного інтервалу часу.

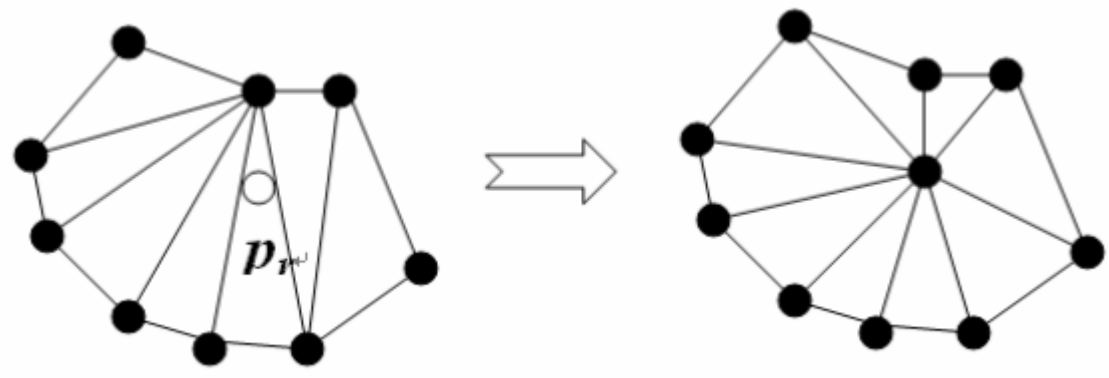


Рис. 3.6 Схематичне зображення перебудови топології при добавлені вузла.

Висновки по розділу

При представленні мережевої топології в вигляді графу, для вирішення мережевих проблем можливо застосовувати геометричні алгоритми на графах. Важливим фактором при цьому являється вибір методу позиціонування вузлів (вибір метрики простору для побудови графу).

Запропонована алгоритмічна схема (фреймворк) по усуненню завад зв'язку в графі з застосуванням алгоритму вимітання площини, тріангуляції Делоне та діаграми Вороного.

РОЗДІЛ 4
ДОСЛІДЖЕННЯ РОБОТИ ЗАПРОПОНОВАНИХ СХЕМ НА ОСНОВІ ГЕОМЕТРИЧНИХ АЛГОРИТМІВ ЗА ДОПОМОГОЮ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

4.1 Дослідження роботи протоколу HWMP в середовищі NS-3.

Для дослідження роботи протоколу HWMP використовувалось комп’ютерне моделювання рухомої мережі передачі даних (випадок використання камер на дронах, Рис. 4.1) використовувалося середовище NS-3 [32]. Саме середовище запускалось на комп’ютері за допомогою Docker-контейнерів; Dockerfile контейнеру наведений в Додатку А

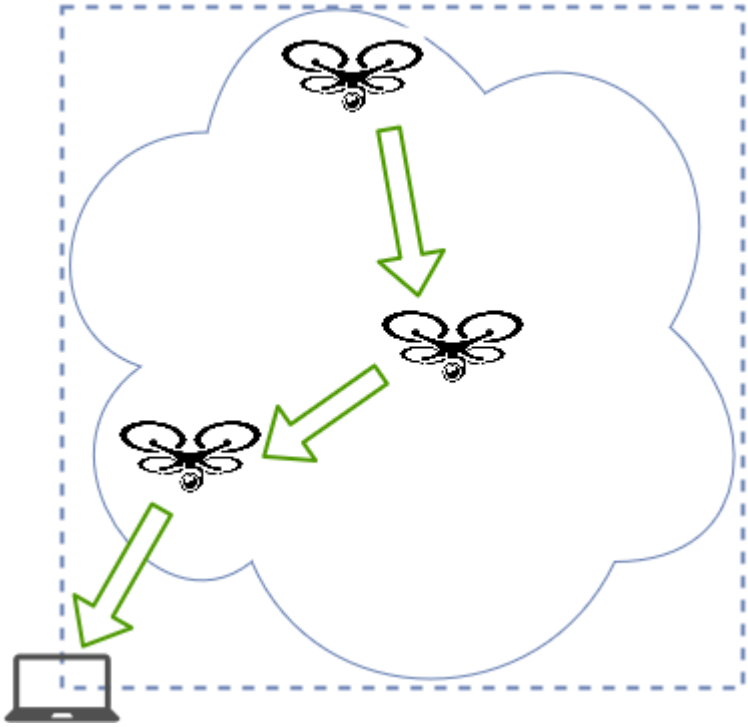


Рис. 4.1. Модель передачі даних від дрону до центру керування та моніторингу.

Код програми, яка використовувалась для моделювання наведений в Додатку Б.

Для аналізу мережевих пакетів, які передавались між дронами використовувалася програма Wireshark [3].

Вузли використовували стандарт зв’язку 802.11n на 5ГГЦ та модель затухання, що описується наступною формулою:

$$\frac{P_r}{P_t} = G_r G_t \left(\frac{\lambda}{4\pi R}\right) \quad (4.1)$$

, де G_r – коефіцієнт підсилення антени-приймача,

G_t – коефіцієнт підсилення антени-передавача,

R – відстань між антенами,

P_t – потужність антени-передавача,

P_r – потужність антени-приймача,

λ – довжина хвилі, що відповідає частоті передачі.

Відстань прямого радіозв'язку була обмежена в 200 метрів. Довжина сторони квадрату периметра змінювалась від 250 до 750 метрів, таким чином змінювалась густина БПЛА в досліджуваній зоні, та маршрути будувались в нових умовах.

В якості метрики якості обслуговування використовувався коефіцієнт PDR (коефіцієнт доставки):

$$PDR = \frac{Rx}{Tx} \quad (4.2)$$

, де Rx – кількість пакетів прийнятих адресатом, Tx – кількість пакетів відправлених джерелом.

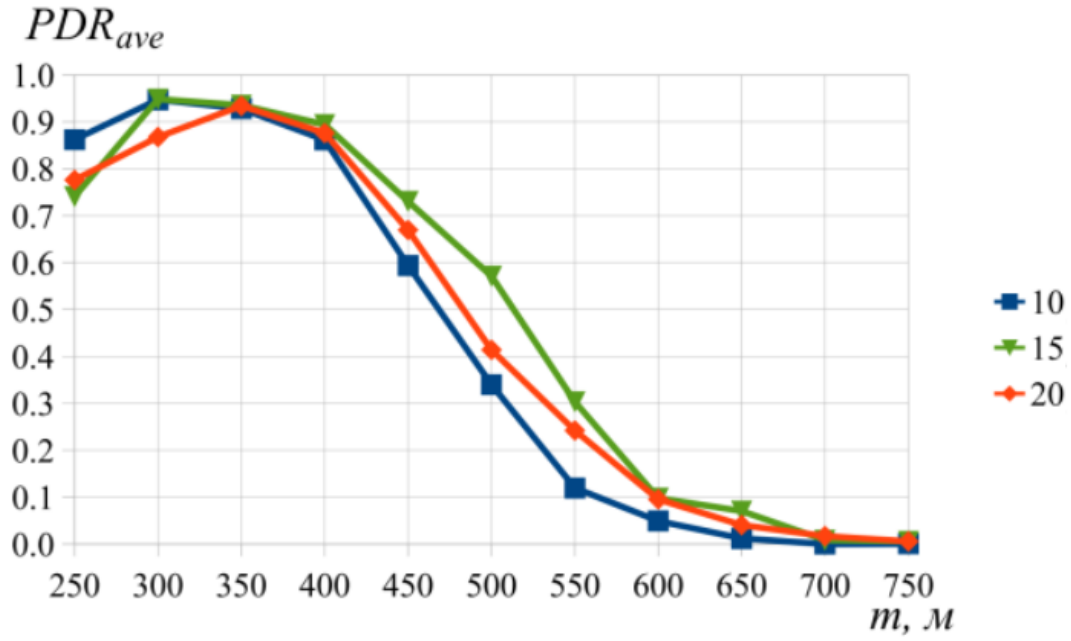


Рис. 4.2 – Середній коефіцієнт доставки (по 3 симуляцій для кожної точки).

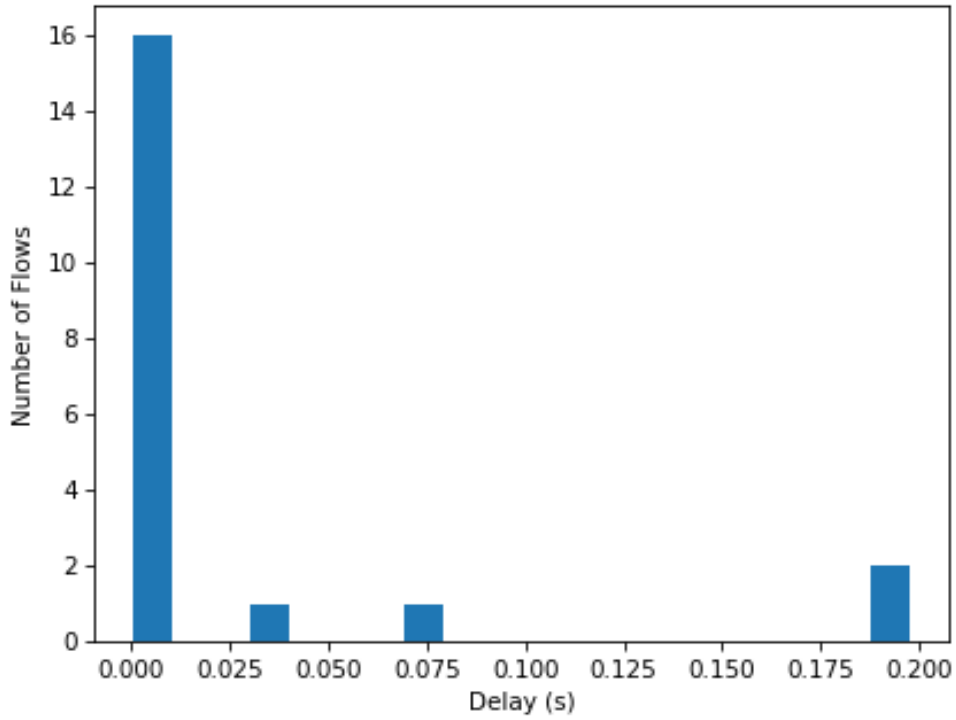


Рис. 4.3 Середній час затримки передачі даних для кожного з БПЛА, при кількості 20; співвідношення дистанції зв’язку до розмірів периметру – 0.5

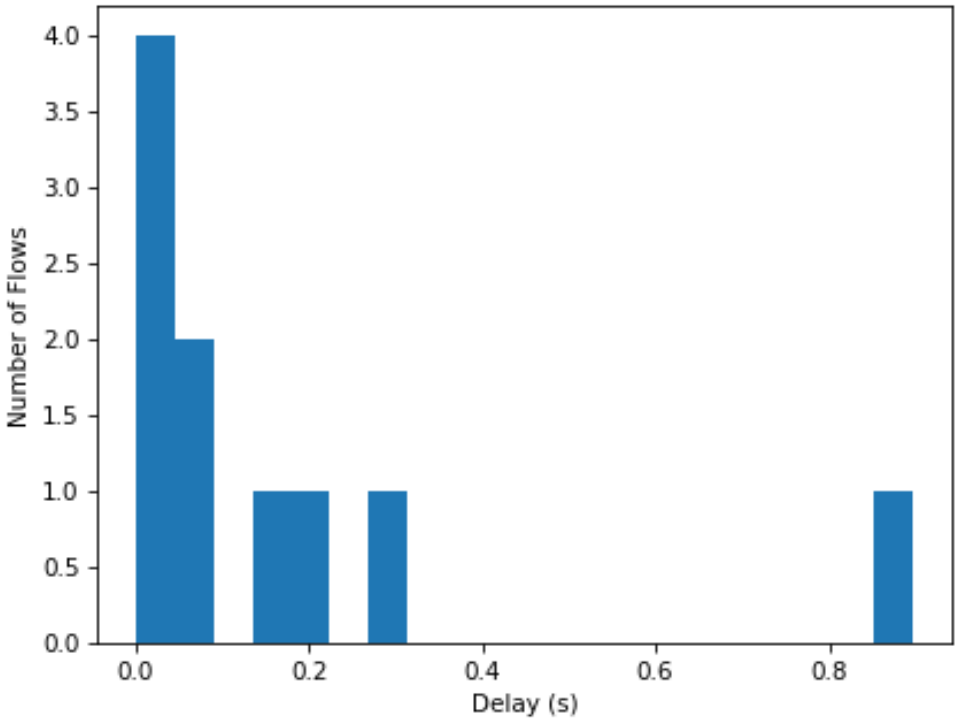


Рис. 4.4 Середній час затримки передачі даних для кожного з БПЛА при кількості 10; співвідношення дистанції зв’язку до розмірів периметру – 0.5

4.2 Параметри середовища комп’ютерного моделювання

Для моделювання мережевих топологій, які змінюються за допомогою запропонованої алгоритмічної структури на основі геометричних алгоритмів необхідно виконати підготовчий процес. Підготовчий процес моделювання: Python використовувався для генерування по 5 різних випадкових розміщень вузлів для 50 та 100 вузлів-передатчиків. Кожен передатчик обмежений дальністю передачі в 300 метрів. Потім використовуємо алгоритм діаграми Вороного для розмежування регіонів серед вузлів. Результат тріангулюємо за допомогою алгоритму тріангуляції Делона, після чого обрізаємо довші зв’язки за результатами застосування алгоритму стандартного відхилення. Дані параметри середовища моделювання наведені в таблиці 4.1

Таблиця 4.1

Параметр	Опис
Мережеве середовище	1) Випадкове розміщення вузлів 2) Розміри об'єкту 1000 (м) x 1000 (м) 3) Кількість вузлів: 50, 100
Властивості вузлів	1) Максимальний діапазон передачі – 300 м. 2) Можливість змінювати параметри передатчика 3) Всебічнонаправлена антена
Інструменти підготовки моделювання	Мова програмування Python відповідальна за: 1) Випадкове розміщення 2) Алгоритм вимітання площини 3) Алгоритм діаграми Вороного 4) Алгоритм тріангуляції Делоне 5) Встановлення з'єднання за протоколом TCP
Мережевий симулятор	NS-3

Таблиця 4.1 Параметри середовища моделювання

4.3Графічне представлення модифікацій графу мережевої топології.

На Рис.4.5 зображена початкова (базова) топологія для сценарію №1 у випадку 100 вузлів-передатчиків із зв'язками в рамках їх дальності передачі.

На Рис.4.6 зображена діаграма Вороного, що була побудована для сценарію №1 зображеного на Рис. 4.5.

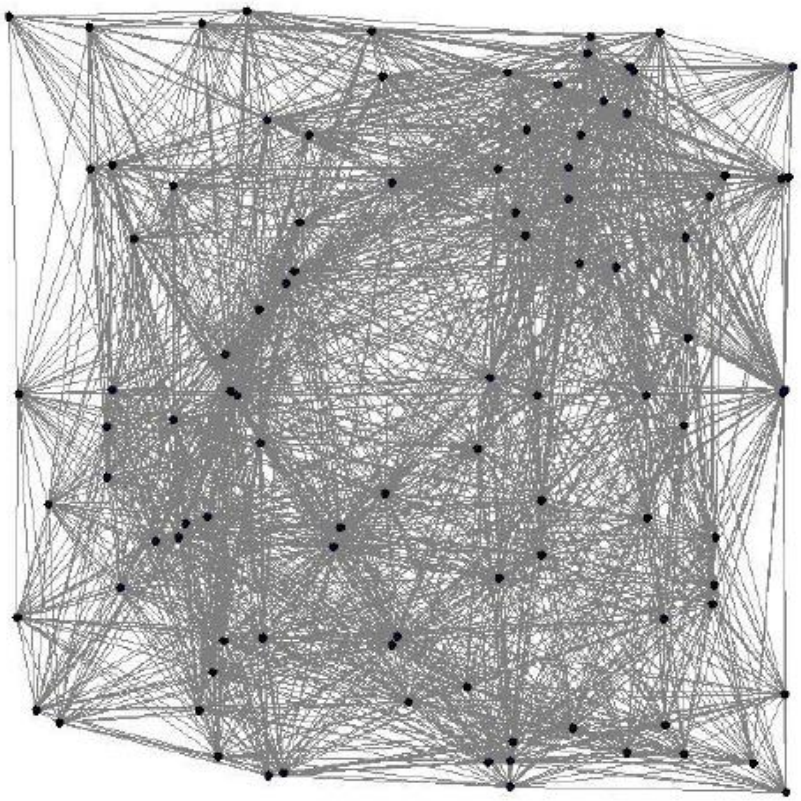


Рис. 4.5 Результат генерування випадкового розміщення вузлів-передатчиків для випадку 100 вузлів.

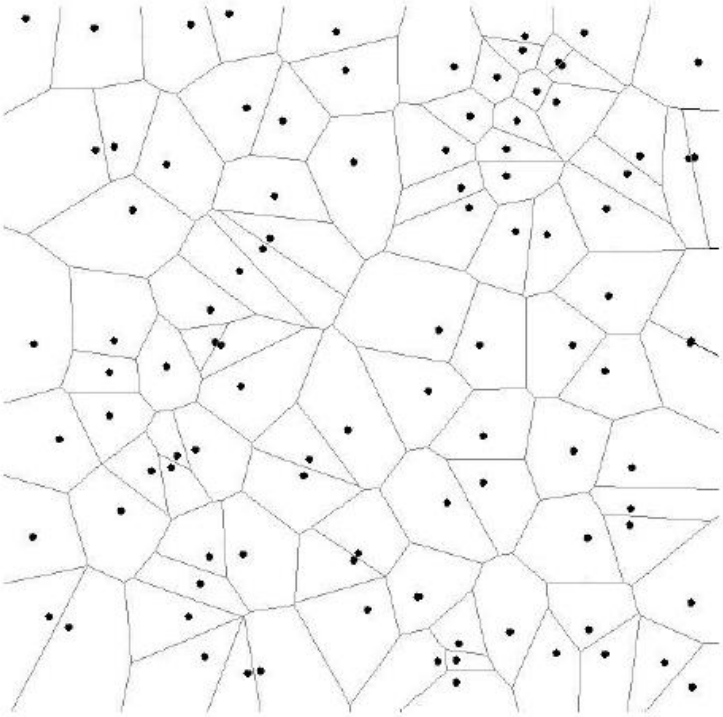


Рис. 4.6 Результат розмежування простору (діаграма Вороного)

Топологію моделі №1 отримуємо за допомогою застосування тріангуляції Делоне до початкової. Графічне представлення зображене на Рис.4.7

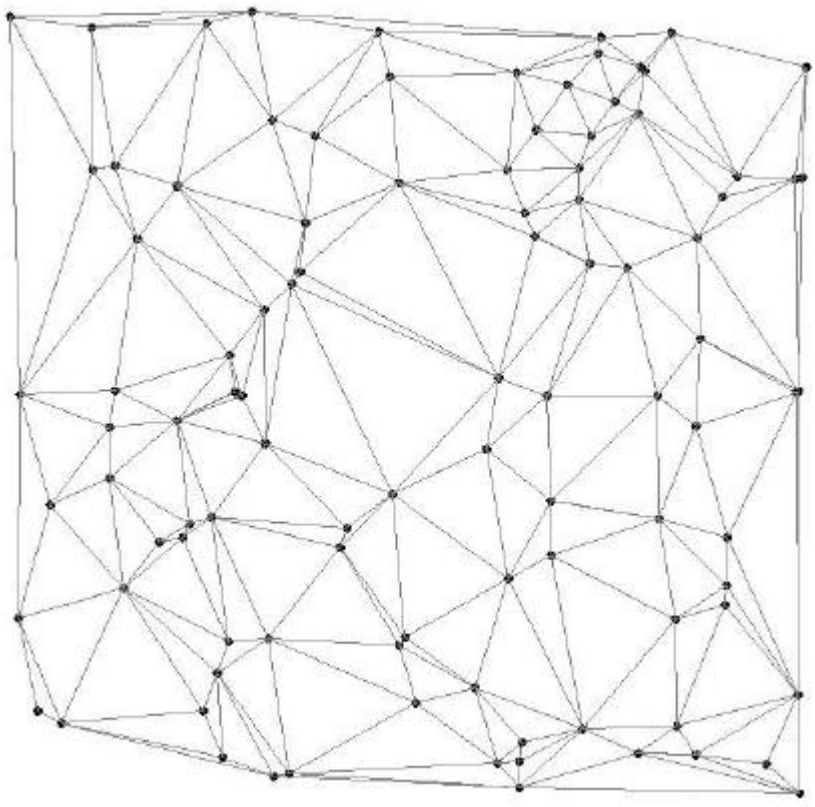


Рис. 4.7. Топологія після застосування тріангуляції Делоне

Складність виконання такої операції на пряму залежить від кількості вузлів графу, n та становить $O(n \cdot \log n)$.

Топологія (Тестова модель) №2, отримується з топології №1 додатковим відкидання неоправдано довгих звязків. Оцінка виконується за допомогою алгоритму стандартного відхилення.

Розглянемо Таблицю 4.2 довжини зв'язків для вузла з ступенем 9 після застосування тріангуляції (9 зв'язків з іншими вузлами):

Таблиця 4.2

Ребро	Довжина ребра	Рівень відхилення (μ = 80.9)
1	12	0 (<μ)
2	250	3(>μ +2σ)
3	22	0 (<μ)
4	210	2(<μ +2σ)
5	36	0 (<μ)
6	30	0 (<μ)
7	33	0 (<μ)
8	120	1(<μ +σ)
9	15	0 (<μ)

Таблиця 4.2 Приклад визначення рівня стандартного відхилення для вузла з ступенем 9

Для цього вузла буде відкидуватись Ребро 2, та Ребро 4, якщо для кінцевого вузла це ребро також має рівень відхилення 2, або вище.

Графічний результат відкидання ребер зображено на Рис. 4.8

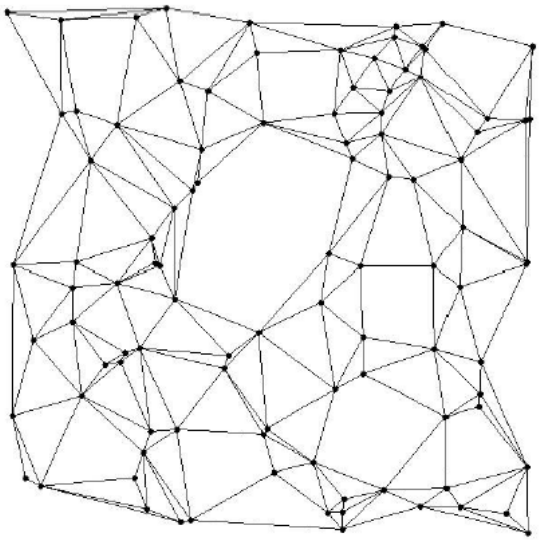
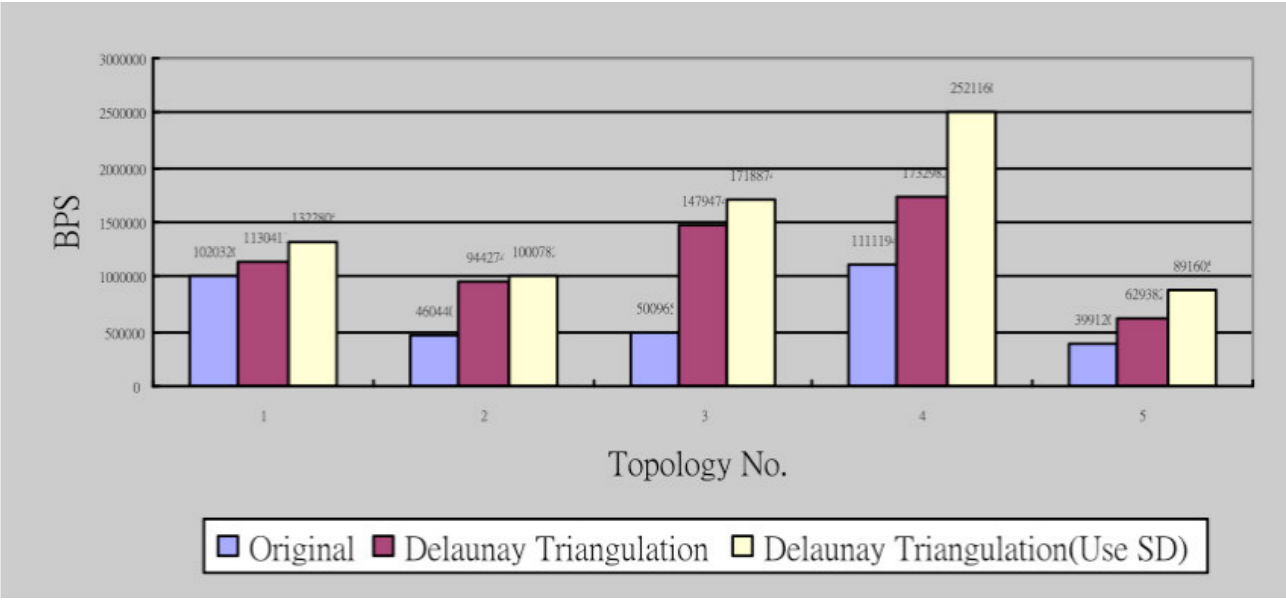


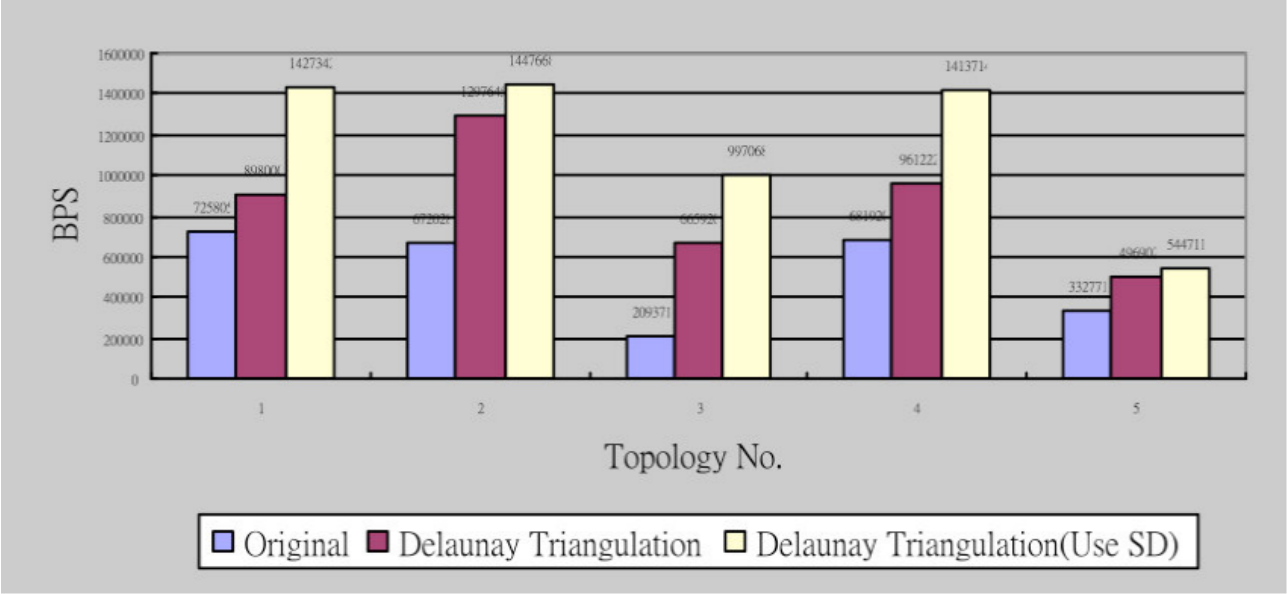
Рис. 4.8 Топлогія після відкидання ребер (стандартне відхилення)

4.4 Порівняння результатів пропускної спроможності.

У цій категорії симуляцій виконувалось порівняння пропускної спроможності "Базового", "Тріангуляції Делоне" та "Тріангуляція Делоне + стандарте відхилення" методів в наступних сценаріях: "50 вузлів та 30 вхідних з'єднань", "100 вузлів та 50 вхідних з'єднань". Для кожного сценарію генерувалось 5 різних топологій. На Рисунку 4.9 зображені результати для сценарію "50 вузлів та 30 вхідних з'єднань". На Рисунку 4.10 зображені результати для "100 вузлів та 50 вхідних з'єднань".



Рисунку 4.9 Порівняння пропускної спроможності ("50 вузлів та 30 вхідних з'єднань")



Рисунку 4.10 Порівняння пропускної спроможності ("100 вузлів та 50 вхідних з'єднань")

По отриманих результатах, спостерігаємо, що метод з використанням "Тріангуляції Делоне" в середньому на 85% краще ніж "Базовий" метод. В свою чергу додаткова модифікація топології за допомогою середнього відхилення ще покращує результат на 35% в середньому.

4.5 Порівняння результатів якості зв'язку.

Під метрикою якості, використовуємо коефіцієнт доставки, який пов'язаний з коефіцієнтом втрат наступним:

$$K_{Quality} = 1 - K_{loss} = 1 - \frac{Tx - Rx}{Tx} \quad (4.3)$$

, де Rx – кількість пакетів прийнятих адресатом, Tx – кількість пакетів відправлених джерелом.

На Рисунку 4.11 зображена діаграма коефіцієнтів втрат для 5 топологій сценарію "50 вузлів та 30 вхідних з'єднань". На Рисунку 4.12 – діаграма коефіцієнтів втрат відповідних 5 топлогій для "100 вузлів та 50 вхідних з'єднань".

Рисунок 4.11 Коефіцієнт втрат ("50 вузлів та 30 вхідних з'єднань")

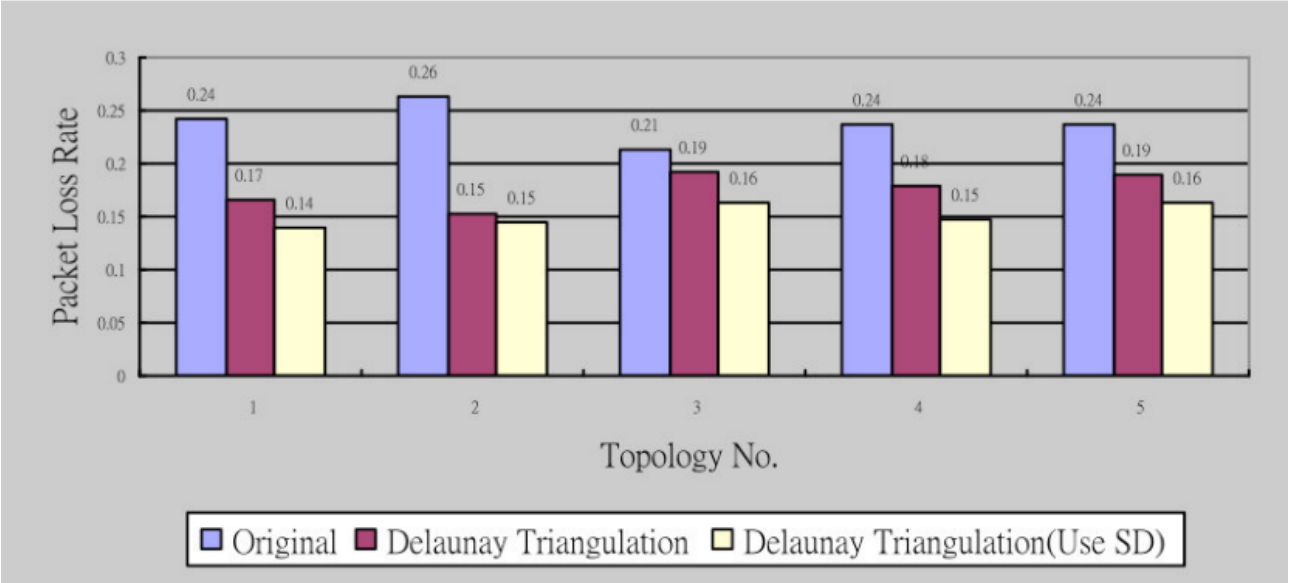


Рисунок 4.12 Коефіцієнт втрат ("100 вузлів та 50 вхідних з'єднань")

4.6 Порівняння часу затримки для встановлення з'єднання

Під часом встановлення з'єднання будемо вважати час, за який виконується TCP-handshake. Метрика часу знімалась на основі часових міток TCP-пакетів отриманих за допомогою програми аналізу мережевих пакетів Wireshark[3]. На Рисунку 4.13 зображена діаграма затримки часу для 5 топологій сценарію "50 вузлів та 30 вхідних з'єднань". На Рисунку 4.14 – діаграма затримки відповідних 5 топлогій для "100 вузлів та 50 вхідних з'єднань".

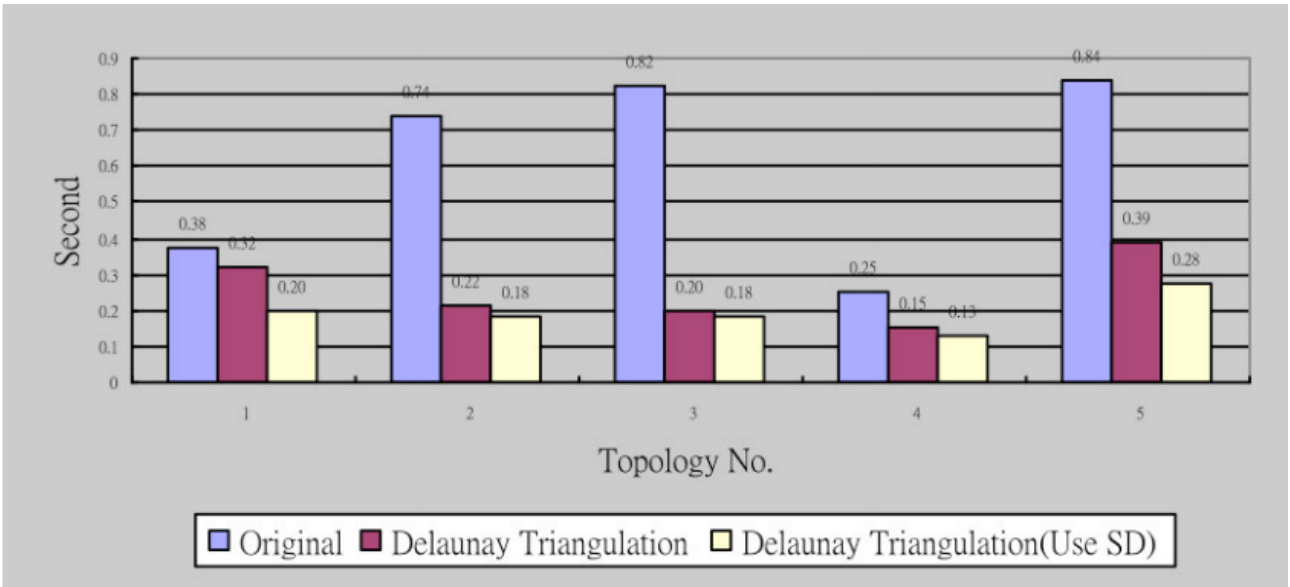


Рисунок 4.13 ("50 вузлів та 30 вхідних з'єднань")

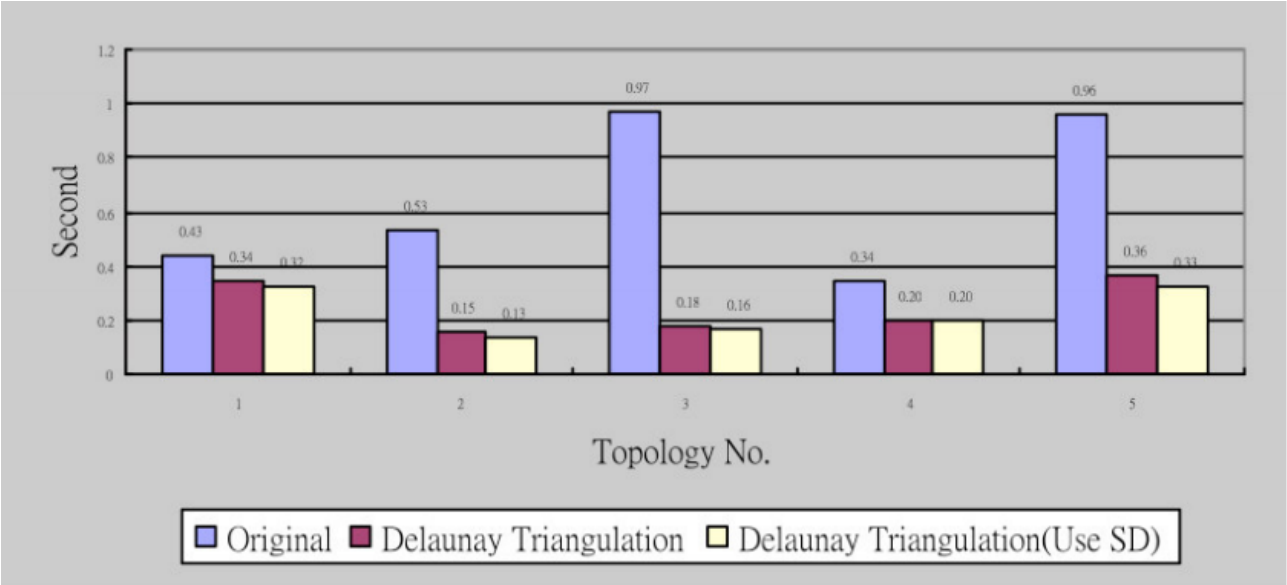


Рисунок 4.14 ("100 вузлів та 50 вхідних з'єднань")

Висновки по розділу

Застосування базової топології з протоколом маршрутизації HWMP, не завжди надає достатню якість з’єднання для цілістної передачі відеоінформації в HD.

Додаткове застосування геометричних алгоритмів є ефективним засобом у збільшенні пропускної здатності, зменшенні обох показників втрат пакетів та часу затримки в середньому.

ВИСНОВКИ

У цій роботі пропонується основа, що використовує геометричні алгоритми, щоб значною мірою зменшити перешкоди між вузлами бездротової мережі. Спочатку ми перетворюємо мережеві задачі на задачі геометрії в теорії графів, а потім вирішуємо задачу перешкод за допомогою геометричних алгоритмів. Спочатку ми визначаємо перетин лінії на графіку, щоб відобразити проблему радіоперешкод у мережі бездротової мережі. Потім ми використовуємо алгоритм зчитування плану, щоб знайти лінії перетину, якщо такі є; використовувати алгоритм діаграми Вороного для розмежування регіонів серед вузлів; використовувати Делоне Алгоритм тріангуляції для реконструкції графіка, щоб мінімізувати перешкоди між вузлами. Нарешті, ми використовуємо стандартне відхилення, щоб відрізати ці довші ланки (більш високі перешкоди) для подальшого покращення. Доведено, що це гібридне рішення здатне значно зменшити перешкоди за час O (n log n).Моделювання показують, що запропонований фреймворк є ефективним у збільшенні пропускної здатності, зменшенні обох показників втрат пакетів та часу затримки в середньому

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. НД ТЗІ 1.6-005-2013 Захист інформації на об'єктах інформаційної діяльності. Положення про категоріювання об’єктів, де циркулює інформація з обмеженим доступом, що не становить державної таємниці.
2. Пшоннік В.О. Охоронні системи швидкого розгортання. Сучасний стан і тенденції розвитку. / Сучасний захист інформації №1(33), 2018 ISSN 2409-7292 – режим доступу до тексту в електронному вигляді: http://journals.dut.edu.ua/index.php/dataprotect/article/view/1795
3. Наказ №906 "Про затвердження Інструкції з технічного забезпечення засобами технічного захисту інформації і комплексами технічного контролю Національної гвардії України" від 27.07.2015
4. Охрана периметра // Концепции безопасности [Електронний ресурс] – режим доступу до ресурсу: http://kb-sb.ru/pub/10/342/
5. Chappell L., Wireshark Network Analysis (Second Edition): The Official Wireshark Certified Network Analyst Study Guide.// ISBN 10: 1893939944 ISBN 13: 9781893939943 Laura Chappell University, 2012
6. Магауенов Р.Г., Системы охранной сигнализации: основы теории и принципы построения, Москва, Горячая линия –Телеком, 2008.
7. Веремеев В. Применение разведывательно-сигнализационных приборов ВС США на современном этапе / В. Веремеев // Зарубежное военное обозрение. – 2017. – №3. – С.51-53
8. Быстроразвертываемые охранные системы [Електронний ресурс] – режим доступу до ресурсу: http://www.gsm-guard.net/press3_1.html
9. Максеменков А. Система разведывательно-сигнализационных приборов "Рембасс" / А. Максеменков. // Зарубежное военное обозрение. – 2006. – №5. – С. 20.
10. Rapid Deployment CCTV Tower // Wireless CCTV Ltd [Електронний ресурс] – режим доступу до ресурсу: https://www.wcctv.co.uk/rapid-deployment-site-security-tower/

11. Rapid Deployment Towers by Tantri // Production overview, 2016 [Електронний ресурс] – режим доступу до ресурсу: https://www.towerxchange.com/wp-content/uploads/2016/09/COW-2.pdf
12. A Security Analysis of the 802.11s Wireless Mesh Network Routing Protocol and Its Secure Routing Protocols [Електронний ресурс]. – режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3821297/
13. A Security Analysis of the 802.11s Wireless Mesh Network Routing Protocol and Its Secure Routing Protocols [Електронний ресурс]. – режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3821297/
14. Ghumman S.A. Per-Arne-Wiberg. Security in Wireless Mesh Networks // School of Information Science, Computer and Electrical Engineering, Halmstad University. – 2009. – № 1. – P . 1–54.
15. Вишневский В . М ., Гузаков Н . Н ., Лаконцев Д . В . Mesh- сети стандарта IEEE 802.11s: протоколы маршрутизации // Первая миля . – 2009. – № 1. – С. 16–21.
16. Zapata M.G. Mobile Ad Hoc Networking Working Group INTERNET DRAFT Secure Ad Hoc On-Demand Distance Vector(SAODV) [Електронний ресурс] – 2013 – режим доступу до ресурсу: http://people.ac.upc.edu/guerrero/papers/draft-guerrero-manet-saodv-06.txt
17. F.I., Wang X., Wang W. Wireless mesh networks: a survey. Computer Networks and ISDN System. – 2005– № 1, p. 445
18. Islam M.S., Hamid M.A., Hong C.S. SHWMP: a secure hybrid wireless mesh protocol for ieee 802.11s wireless mesh network. IEEE Transactions on Computers in Science, 2009, № 6, pp. 95-114.
19. Yi P., Wu Y., Zou F., Liu N. A survey on security in wireless mesh networks. IETE Tech. Rev, 2010, no. 27, pp. 6-14
20. Al-Shurman M., Yoo S.M., Park S. Black Hole Attack in Wireless Ad Hoc Networks. Proceedings of ACM 42nd Southeast Conference, Huntsville, AL, USA, 2004, pp. 1-3.

21. Khan S., Mast N., Loo K.K., Salahuddin A. Passive Security Threats and Consequences in IEEE 802.11 Wireless Mesh Networks. Int. J. Dig. Cont. Tech, 2008, pp. 4-8.
22. Naeem T., Loo K.K. Common Security Issues and Challenges in Wireless Sensor Networks and IEEE 802.11 Wireless Mesh Networks. Int. J. Dig. Cont. Tech , 2009, pp. 88-93.
23. Mikrotik Documentation: Interface/HWMPplus [Електронний ресурс] – режим доступу до ресурсу: https://wiki.mikrotik.com/wiki/Manual:Interface/HWMPplus
24. 802.11 Working Group of the IEEE 802 Committee. IEEE P802.11s . D4.01 Draft Standard. IEEE, Washington, DC, USA, 2010. [Електронний ресурс] – режим доступу до ресурсу: http://www.ieee802.org/11/Reports/tgd_update.htm
25. Clausen T., Dearlove C., Jacquet P. The Optimized Link State Protocol version 2 // IETF draft, September 2009 [Електронний ресурс] – режим доступу до ресурсу: https://tools.ietf.org/html/draft-ietf-manet-olsrv2-10
26. Neumann, A. et al. Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N.) // IETF draft, April 2008 [Електронний ресурс] – режим доступу до ресурсу: https://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00
27. Wikipedia: Destination-Sequenced Distance Vector routing [Електронний ресурс] – режим доступу до ресурсу: https://en.wikipedia.org/wiki/Destination-Sequenced_Distance_Vector_routing
28. Perkins C., Belding-Royer E., Das S. Ad hoc On-Demand Distance Vector (AODV) Routing. // RFC3561, July 2003[Електронний ресурс] – режим доступу до ресурсу: https://www.ietf.org/rfc/rfc3561.txt
29. Johnson D., Hu Y., Maltz D. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4 //RFC4728, February 2007 [Електронний ресурс] – режим доступу до ресурсу: https://www.ietf.org/rfc/rfc4728.txt

30. Chroboczek J., The Babel Routing Protocol // RFC6126, April 2011 [Електронний ресурс] – режим доступу до ресурсу: https://www.ietf.org/rfc/rfc6126.txt
31. R.P. Draves Jr., B. D. Zill, J.D. Padhye System and Method for Link Quality Source Routing // U.S. Patent 7376122
32. Haas, Pearlman, Samar Zone Routing Protocol (ZRP) for Ad Hoc Networks // IETF draft, July 2002 [Електронний ресурс] – режим доступу до ресурсу: https://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04
33. Pei G., Gerla M., Tsu-Wei C. Fisheye State Routing in Mobile Ad Hoc Networks // Conference Paper, 2000 [Електронний ресурс] – режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.6730
34. NS-3: A Discreate-Event Network Simulator // [Електронний ресурс] – режим доступу: https://www.nsnam.org/docs/manual/html/index.html
35. MCS 481 Lecture 26 Computational Geometry Jan Verschelde, 2019 // [Електронний ресурс] – режим доступу: http://homepages.math.uic.edu/~jan/mcs481/index.html

ДОДАТОК А NS3 DOCKERFILE

```
FROM ryankurte/docker-ns3:latest

RUN apt-get update && \
apt-get install -y \
nano \
qt5-default \
python-pygraphviz \
python-kiwi \
python-pygoocanvas \
libgoocanvas-dev \
ipython \
openmpi-bin \
openmpi-common \
openmpi-doc \
libopenmpi-dev \
uncrustify \
doxygen \
imagemagick \
texlive \
texlive-extra-utils \
texlive-latex-extra \
texlive-font-utils \
texlive-lang-portuguese \
dvipng \
```

Продовження додатку А

```
RUN apt-get -y install wget
    RUN cd /tmp && \
    wget https://bootstrap.pypa.io/get-pip.py && \
    python get-pip.py && \
    apt-get install -y gccxml python-pygccxml python-gnome2 python-rsvg


    RUN pip install matplotlib
    ADD manet_routing_compare.py /usr/ns3/ns-3.26/
    WORKDIR /usr/ns3/ns-3.26
    VOLUME ["/usr/ns3/ns-3.26/work"]

   CMD ["./waf", "--pyrun", "routing_test.py"]
```

ДОДАТОК Б КОД СИМУЛЯЦІЇ ФАЙЛУ ROUTING_TEST.PY

```
import csv
import sys
import os
```

Продовження додатку Б

```
import time

import matplotlib

# Force matplotlib to not use any Xwindows backend.
```

```
matplotlib.use('Agg')

import matplotlib.pyplot as plt
import ns.aodv
import ns.applications
import ns.flow_monitor
import ns.core
import ns.dsdv
import ns.dsr
import ns.internet
import ns.mobility
import ns.network
import ns.olsr
import ns.wifi

__location__ = os.path.realpath(
    os.path.join(os.getcwd(), os.path.dirname(__file__)))
__workdir__ = os.path.join(__location__, "work")


class RoutingExperiment:
    def __init__(self):
```

Продовження додатку Б

Продовження додатку Б

```
        # Refresh on run time
        self.bytesTotal = 0
        self.packetsReceived = 0
        # Default configs
        self.port = 9
```

```
        self.m_CSVfileName = "mesh-routing.output"
        self.m_nodes = 50
        self.m_protocolName = ""
        self.m_txp = 8.9048
        self.m_total_time = 1000
        self.m_node_speed = 20  # in m/s
        self.m_debugger = True

        # Used to simulations
        self.m_nSinks = 10
        self.m_protocol = 3
        self.m_node_pause = 0  # in s

    # Ptr<Socket> socket, Ptr<Packet> packet, Address senderAddress
    # returns string
    def PrintReceivedPacket(self, socket, packet, senderAddress):
        oss = str(ns.core.Simulator.Now().GetSeconds()) + " " +
str(socket.GetNode().GetId())

        if ns.network.InetSocketAddress.IsMatchingType(senderAddress):
            addr = ns.network.InetSocketAddress.ConvertFrom(senderAddress) #
type: InetSocketAddress
            ipv4 = addr.GetIpv4()  # type: Ipv4Address
```

Продовження додатку Б

```
            sys.stdout.write(oss)
            print ipv4
        else:
            print oss
```

```
        # Ptr<Socket> socket
        # returns void
    def ReceivePacket(self, socket):
            senderAddress = ns.network.Address()  # type: ns.network.Address
            packet = socket.RecvFrom(senderAddress)
            while packet != None:
                self.bytesTotal += packet.GetSize()
                self.packetsReceived += 1
                self.PrintReceivedPacket(socket, packet, senderAddress)
                packet = socket.RecvFrom(senderAddress)


        def WriteHeaderCsv(self)
            with open(os.path.join(__workdir__, (self.m_CSVfileName + ".csv")), 'w')
as csvfile:
                spamwriter = csv.writer(csvfile, delimiter=';',
                                        quotechar='|', quoting=csv.QUOTE_MINIMAL)
                spamwriter.writerow(
                    ['SimulationSecond',
                     'ReceiveRate',
                     'PacketsReceived',
                     'PacketDeliveryRatio',
                     'NumberOfSinks',
```

Продовження додатку Б

Продовження додатку Б

```
     'RoutingProtocol',
                     'TransmissionPower'])


        def CheckThroughput(self):
            kbs = (self.bytesTotal * 8.0) / 1000
```

```
            now = int((ns.core.Simulator.Now()).GetSeconds())

            # Packet delivery ratio

        # 4 is number of packets send each second

            pdr = self.packetsReceived / (4 * self.m_nSinks)
            with open(os.path.join(__workdir__, (self.m_CSVfileName + ".csv")), 'a') as csvfile:
                spamwriter = csv.writer(csvfile, delimiter=';',
                                        quotechar='|', quoting=csv.QUOTE_MINIMAL)
                spamwriter.writerow([now, kbs, self.packetsReceived, pdr, self.m_nSinks, self.m_protocolName, self.m_txp])

            self.packetsReceived = 0
            ns.core.Simulator.Schedule(ns.core.Seconds(1.0), RoutingExperiment.CheckThroughput, self)

        # Ipv4Address addr, Ptr<Node> node
        # returns Ptr<Socket>
        def SetupPacketReceive(self, addr, node):
            tid = ns.core.TypeId.LookupByName("ns3::UdpSocketFactory")  # type: TypeId
```

Продовження додатку Б

```
            sink = ns.network.Socket.CreateSocket(node, tid)  # type: Ptr<Socket>
```

Продовження додатку Б

```
        local = ns.network.InetSocketAddress(addr, self.port)  # type: InetSocketAddress
            sink.SetRecvCallback(self.ReceivePacket)
```

```
        return sink

    @staticmethod
    def print_stats(output, st):
        print >> output, "  Tx Bytes: ", st.txBytes
        print >> output, "  Rx Bytes: ", st.rxBytes
        print >> output, "  Tx Packets: ", st.txPackets
        print >> output, "  Rx Packets: ", st.rxPackets
        print >> output, "  Lost Packets: ", st.lostPackets
        if st.rxPackets > 0:
            print >> output, "  Mean{Delay}: ", (st.delaySum.GetSeconds() / st.rxPackets)
            print >> output, "  Mean{Jitter}: ", (st.jitterSum.GetSeconds() / (st.rxPackets - 1))
            print >> output, "  Mean{Hop Count}: ", float(st.timesForwarded) / st.rxPackets + 1

            print >> output, "Delay Histogram"
            for i in range(st.delayHistogram.GetNBins()):
                print >> output, " ", i, "(", st.delayHistogram.GetBinStart(i), "-", \
                    st.delayHistogram.GetBinEnd(i), "): ", st.delayHistogram.GetBinCount(i)
```

Продовження додатку Б

```
            for i in range(st.jitterHistogram.GetNBins()):
                print >> output, " ", i, "(", st.jitterHistogram.GetBinStart(i), "-", \
                    st.jitterHistogram.GetBinEnd(i), "): ", st.jitterHistogram.GetBinCount(i)
            print >> output, "PacketSize Histogram"
            for i in range(st.packetSizeHistogram.GetNBins()):
```

```
                print >> output, " ", i, "(", st.packetSizeHistogram.GetBinStart(i), "-", \
                    st.packetSizeHistogram.GetBinEnd(i), "): ",
st.packetSizeHistogram.GetBinCount(i)
        for reason, drops in enumerate(st.packetsDropped):
            print "  Packets dropped by reason %i: %i" % (reason, drops)
            # for reason, drops in enumerate(st.bytesDropped):

    #    print "Bytes dropped by reason %i: %i" % (reason, drops)
     # int nSinks, double txp, std::string CSVfileName
     def Run(self, *positional_parameters, **keyword_parameters):
        ns.network.Packet.EnablePrinting()
        if "SINKS" in os.environ:
            self.m_nSinks = int(os.environ["SINKS"])
        if "TXP" in os.environ:
            self.m_txp = float(os.environ["TXP"])
        if "TOTAL_TIME" in os.environ:
            self.m_total_time = int(os.environ["TOTAL_TIME"])
        if "NODES" in os.environ:
            self.m_nodes = int(os.environ["NODES"])
        if "PROTOCOL" in os.environ:
```

Продовження додатку Б

```
            self.m_protocol = int(os.environ["PROTOCOL"])
```

Продовження додатку Б

```
        if "NODE_SPEED" in os.environ:
            self.m_node_speed = int(os.environ["NODE_SPEED"])
        if "NODE_PAUSE" in os.environ:
```

Продовження додатку Б

```
            self.m_node_pause = int(os.environ["NODE_PAUSE"])
```

```
        if "FILE_NAME" in os.environ:
            self.m_CSVfileName = os.environ["FILE_NAME"]

        self.m_CSVfileName += "." + str(time.time())

    rate = "2048bps"  # 4 packets/s => 64bytes * 8 = 512bits * 4 = 2048 bits por segundo
        phyMode = "DsssRate11Mbps"
        tr_name = self.m_CSVfileName + "-compare"
        self.m_protocolName = "protocol"
        ns.core.Config.SetDefault("ns3::OnOffApplication::PacketSize", ns.core.StringValue("64"))
    ns.core.Config.SetDefault("ns3::OnOffApplication::DataRate", ns.core.StringValue(rate))

        # Set Non-unicastMode rate to unicast mode


ns.core.Config.SetDefault("ns3::WifiRemoteStationManager::NonUnicastMode", ns.core.StringValue(phyMode));

        adhocNodes = ns.network.NodeContainer()
        adhocNodes.Create(self.m_nodes)
```

Продовження додатку Б

```
        # setting up wifi phy and channel using helpers
```

Продовження додатку Б

```
      wifi = ns.wifi.WifiHelper()
        wifi.SetStandard(ns.wifi.WIFI_PHY_STANDARD_80211n)
```

```
        wifiPhy = ns.wifi.YansWifiPhyHelper.Default()
        wifiChannel = ns.wifi.YansWifiChannelHelper()

wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel")
        wifiChannel.AddPropagationLoss("ns3::FriisPropagationLossModel")
        wifiPhy.SetChannel(wifiChannel.Create())
        # Add a mac and disable rate control
        wifiMac = ns.wifi.WifiMacHelper()
        wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager",
                                     "DataMode", ns.core.StringValue(phyMode),
                                     "ControlMode", ns.core.StringValue(phyMode))

      wifiPhy.Set("TxPowerStart", ns.core.DoubleValue(self.m_txp))
        wifiPhy.Set("TxPowerEnd", ns.core.DoubleValue(self.m_txp))
        wifiMac.SetType("ns3::AdhocWifiMac")

    adhocDevices = wifi.Install(wifiPhy, wifiMac, adhocNodes)  # type:
NetDeviceContainer

        mobilityAdhoc = ns.mobility.MobilityHelper()
        streamIndex = 0  # used to get consistent mobility across scenarios
        pos = ns.core.ObjectFactory()
```

Продовження додатку Б

```
        pos.SetTypeId("ns3::RandomRectanglePositionAllocator"
```

Продовження додатку Б

```
      pos.Set("X",
ns.core.StringValue("ns3::UniformRandomVariable[Min=0.0|Max=300.0]"))
        pos.Set("Y",
ns.core.StringValue("ns3::UniformRandomVariable[Min=0.0|Max=1500.0]"))
```

```
    # Same as: Ptr<PositionAllocator> taPositionAlloc = pos.Create ()->GetObject<PositionAllocator> ();
        taPositionAlloc = pos.Create().GetObject(
          ns.mobility.PositionAllocator.GetTypeId())          # type: Ptr<PositionAllocator>
        streamIndex += taPositionAlloc.AssignStreams(streamIndex)
        ssSpeed = "ns3::UniformRandomVariable[Min=0.0|Max=%s]" % self.m_node_speed
        ssPause = "ns3::ConstantRandomVariable[Constant=%s]" % self.m_node_pause

mobilityAdhoc.SetMobilityModel("ns3::RandomWaypointMobilityModel",
                               "Speed", ns.core.StringValue(ssSpeed),
                               "Pause", ns.core.StringValue(ssPause),

     "PositionAllocator", ns.core.PointerValue(taPositionAlloc))
        mobilityAdhoc.SetPositionAllocator(taPositionAlloc)
        mobilityAdhoc.Install(adhocNodes)
        streamIndex += mobilityAdhoc.AssignStreams(adhocNodes, streamIndex)
        # NS_UNUSED(streamIndex) # From this point, streamIndex is unused
```

Продовження додатку Б

```
    aodv = ns.aodv.AodvHelper()
    olsr = ns.olsr.OlsrHelper()
        dsdv = ns.dsdv.DsdvHelper()
        dsr = ns.dsr.DsrHelper()
    dsrMain = ns.dsr.DsrMainHelper()
        list = ns.internet.Ipv4ListRoutingHelper()
```

```
        internet = ns.internet.InternetStackHelper()

        if self.m_protocol == 1:
            list.Add(olsr, 100)
            self.m_protocolName = "OLSR"
      elif self.m_protocol == 2:
            list.Add(aodv, 100)
            self.m_protocolName = "AODV"
        elif self.m_protocol == 3:
            list.Add(dsdv, 100)
            self.m_protocolName = "DSDV"
        elif self.m_protocol == 4:
            self.m_protocolName = "DSR"
        else:
            print("No such protocol:%s" % str(self.m_protocol))  # NS_FATAL_ERROR ("No such protocol:" << m_protocol);

        if self.m_protocol < 4:
            internet.SetRoutingHelper(list)
      internet.Install(adhocNodes)
        elif self.m_protocol == 4:
            internet.Install(adhocNodes)
```

Продовження додатку Б

```
        print("assigning ip address")  # NS_LOG_INFO("assigning ip address");

      addressAdhoc = ns.internet.Ipv4AddressHelper()
        addressAdhoc.SetBase(ns.network.Ipv4Address("10.1.1.0"), ns.network.Ipv4Mask("255.255.255.0"))
        adhocInterfaces = addressAdhoc.Assign(adhocDevices)
```

```
            onoff1 = ns.applications.OnOffHelper("ns3::UdpSocketFactory",
ns.network.Address())
          onoff1.SetAttribute("OnTime",
ns.core.StringValue("ns3::ConstantRandomVariable[Constant=1.0]"))
            onoff1.SetAttribute("OffTime",
ns.core.StringValue("ns3::ConstantRandomVariable[Constant=0.0]"))
            for i in range(0, self.m_nSinks):
              # Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress
(i), adhocNodes.Get (i));
              sink = self.SetupPacketReceive(adhocInterfaces.GetAddress(i),
adhocNodes.Get(i))

              remoteAddress = ns.network.AddressValue(
                ns.network.InetSocketAddress(adhocInterfaces.GetAddress(i),
self.port))
              onoff1.SetAttribute("Remote", remoteAddress);

            # Ptr<UniformRandomVariable> var =
CreateObject<UniformRandomVariable> ();
```

Продовження додатку Б

```
        posURV = ns.core.ObjectFactory()
```

Продовження додатку Б

```
              posURV.SetTypeId("ns3::UniformRandomVariable")
              var =
posURV.Create().GetObject(ns.core.UniformRandomVariable.GetTypeId())
              temp = onoff1.Install(adhocNodes.Get(i + self.m_nSinks))  # type:
ApplicationContainer
              temp.Start(ns.core.Seconds(var.GetValue(100.0, 101.0)))
```

```
            temp.Stop(ns.core.Seconds(self.m_total_time))
        ss = self.m_nSinks
            nodes = str(ss)

        ss2 = self.m_node_speed

            sNodeSpeed = str(ss2)

            ss3 = self.m_node_pause
            sNodePause = str(ss3)

            ss4 = rate
            sRate = str(ss4)

            # NS_LOG_INFO ("Configure Tracing.");
            tr_name = tr_name + "_" + \
                    self.m_protocolName + "_" + \
                    nodes + "sinks_" + \
                    sNodeSpeed + "speed_" + \
                    sNodePause + "pause_" + \
                    sRate + "rate"
         self.m_CSVfileName = tr_name
```

Продовження додатку Б

```
        ascii = ns.network.AsciiTraceHelper()
            #
wifiPhy.EnableAsciiAll(ascii.CreateFileStream(os.path.join(__workdir__, "%s.tr" %
tr_name)))
```

```
        ns.mobility.MobilityHelper.EnableAsciiAll(ascii.CreateFileStream(os.path.join(
__workdir__, "%s.mob" % tr_name)))
         # internet.EnablePcapAll("wifi-olsr")
        flowmon_helper = ns.flow_monitor.FlowMonitorHelper()
             #flowmon_helper.SetMonitorAttribute("StartTime",
ns.core.TimeValue(ns.core.Seconds(31)))
        monitor = flowmon_helper.InstallAll()
             monitor = flowmon_helper.GetMonitor()
             monitor.SetAttribute("DelayBinWidth", ns.core.DoubleValue(0.001))
             monitor.SetAttribute("JitterBinWidth", ns.core.DoubleValue(0.001))
             monitor.SetAttribute("PacketSizeBinWidth", ns.core.DoubleValue(20))

             print("Run Simulation.")

             self.WriteHeaderCsv()
        ns.core.Simulator.Stop(ns.core.Seconds(self.m_total_time))
             ns.core.Simulator.Run()

             monitor.CheckForLostPackets()
             classifier = flowmon_helper.GetClassifier()
```

Продовження додатку Б

```
             if self.m_debugger:
                for flow_id, flow_stats in monitor.GetFlowStats():
```

Продовження додатку Б

```
                    with open(os.path.join(__workdir__, (self.m_CSVfileName + ".txt")),
'a') as file:
                      # spamwriter = csv.writer(csvfile, delimiter=';',
                      #                         quotechar='|', quoting=csv.QUOTE_MINIMAL)
         # spamwriter.writerow(
```

Продовження додатку Б

```
[now, kbs, self.packetsReceived, pdr, self.m_nSinks, self.m_protocolName, self.m_txp])
                    t = classifier.FindFlow(flow_id)
        proto = {6: 'TCP', 17: 'UDP'}[t.protocol]
                    file.write("FlowID: %i (%s %s/%s --> %s/%i)\n" % \

                     (flow_id, proto, t.sourceAddress, t.sourcePort, t.destinationAddress, t.destinationPort))
                    self.print_stats(file, flow_stats)
                    file.close()

    monitor.SerializeToXmlFile(os.path.join(__workdir__, "%s.flowmon" % tr_name), True, True)
        delays = []
            for flow_id, flow_stats in monitor.GetFlowStats():
                tupl = classifier.FindFlow(flow_id)
                if tupl.protocol == 17 and tupl.sourcePort == 698:
                    continue
         if flow_stats.rxPackets == 0:
                    delays.append(0)
                else:
```

Продовження додатку Б

```
         delays.append(flow_stats.delaySum.GetSeconds()
```

Продовження додатку Б

```
    flow_stats.rxPackets)
    plt.hist(delays, 20)
            plt.xlabel("Delay (s)")
            plt.ylabel("Number of Flows")
            plt.savefig(os.path.join(__workdir__, '%s.png' % tr_name), dpi=75)
```

```
        plt.show()
    ns.core.Simulator.Destroy()
if __name__ == "__main__":
    v = RoutingExperiment() ruv\\
```