Хмельницький національний університет
Факультет інформаційних технології
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Мультикомп'ютерна система згідно топології «сніжинка»
Назва теми

КвРКІ. 200298.27.07.02 ПЗ
Шифр

Галузь знань 12 «Інформаційні технології»
Шифр, назва

Спеціальність 123 «Комп'ютерна інженерія»
Шифр, назва

Освітня програма «Комп'ютерна інженерія та програмування»
Назва

Виконав: студент IV курсу, група КІ2–19–2 _____ С. В. Коцюрба
Підпис Ініціали, прізвище

Керівник _____ Д. М. Медзатий
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«1» червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т. О. Говорущенко

" 11" 01 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Коцюрба Сергій Валерійович

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мультикомп'ютерна система згідно топології «сніжинка»

Керівник проекту (роботи) Медзатий Д.М, доцент кафедри КІС

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 07.06.2023 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Концепція автономних систем та аналіз відомих засобів та рішень

Проєктування мультикомп'ютерної системи з топологією «сніжинка»

Реалізація мультикомп'ютерної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Схема передачі повідомлень та основних складових частин системи повідомлень

Фрагмент топології

Схема топології

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_11_» 01 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	11.01.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2023	виконано
3	Робота над розділом 1 – Концепція автономних мереж, топології та аналіз відомих засобів та рішень	01.03.2023	виконано
4	Робота над розділом 2 – Вибір топології та проєктування системи	01.04.2023	виконано
5	Робота над розділом 3 – Підходи до побудови системи з використанням кластерів	30.04.2023	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2023	виконано
7	Попередній захист ВКР	26.05.2023	виконано
8	Захист ВКР на засіданні ЕК	Червень 2023 року	

Студент Підпис Кошорба С. В. Ініціали, прізвище

Керівник проекту (роботи) Підпис Медзатин Д. М. Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл-листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ. 200298.27.07.02 ПЗ	Пояснювальна записка	56		
			Графічні матеріали			
2		КвРКІ. 200298.27.07.02 Е8	Схема передачі	1		
			повідомлень та основних			
			складових частин			
			системи повідомлень			
3		КвРКІ. 200298.27.07.02 Е8	Схема топології	1		
4		КвРКІ. 200298.27.07.02 Е8	Фрагмент топології	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ. 200298.27.07.02 ПЗ		
Розробив		Кошурба С.			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Медовий Л.		31.05		У	
Н. контр.		Лисенко С.М.			ХНУ, КІ2-19-2		
Затв.		Гончуковська Т.		31.05			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Мультикомп'ютерна система згідно топології «сніжинка»

Автор роботи: *Коцюрба Сергій Валерійович*.

Керівник роботи: *Медзатий Дмитро Миколайович*.

Пояснювальна записка: 66 *с*., 6 *рис*., 1 *табл*., 3 *дод*., 40 *джерел*.

Графічна частина: 15 презентаційних слайдів.

Ключові слова: мультикомп'ютерна система, топологія, автономна мережа.

Мета роботи - розробка мультикомп'ютерні системи на базі топології «сніжинка».

Об'єктом дослідження є процеси взаємодії між вузлами кластерів.

Предметом дослідження є мультикомп'ютерні системи та протоколи взаємодії і обміну повідомленнями між їх вузлами.

Тема створення мультикомп'ютерні системи на базі топології «сніжинка» є актуальною. Обчислення, які здійснюються з використанням сучасних комп'ютерів та багатомашинних комплексів зростають. В зв'язку з цим зростає потреба у збільшенні обчислювалних ресурсів. Крім того, недостатність стандарту IPv4 спонукатиме перехід до IPv6. Наявність багатьох вузлів в комп'ютерних мережах потребує узгодження їх взаємодії через відповідні протоколи та засоби взаємодії. Все це може бути вирішене створенням мультикомп'ютерних систем для розв'язання певного класу задач. При їх створенні було застосовано топологію «сніжинка», що найбільш адекватно відповідає наявній топології вузлів в мережах. Навколо частини прикінцевих вузлів сформовано кластери, в яких один з вузлів є головою списку вузлів в кластері. Обробка повідомлень в таких кластерах є важливою в контексті фракталів сніжинок. В результаті, було запропоновано використовувати гібридний підхід до створення мультикомп'ютерні системи на базі топології «сніжинка», який враховує як централізацію так і децентралізацію.

31.05.23

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АПУ - Автономна площина управління

AMI - Інфраструктура автономної мережі

ARP - Автономний агент

ААО - служби

GRASP - головний автономний сигнальний протокол

ВСТУП

Управління мережею дозволяє мережі досягти більш високої продуктивності, зниження витрат і досягнення цілей користувачів. Управління мережею надає вузлам можливості для відновлення після збою та оптимізації їх продуктивності. Мережами можна керувати централізовано або розподілено. Зі збільшенням кількості мережевих пристроїв управління стало більш складним завданням. Складність мережі залежить від кількості вузлів та/або кількості ролей, які вони можуть отримати в мережі. Оскільки мережі збільшуються в розмірах, зростає і їх залежність від адміністраторів-людей, так що управління мережею відповідає темпам зростаючої інфраструктури. Оскільки розмір мереж зростає, і вони піддаються впливу інших мереж, шанси зазнати електронних атак та наявності шкідливих вузлів всередині мереж можуть зростати. Крім того, швидкість, з якою мережа відчуває відмову вузлів, також, може зростати. Тому, впровадження рішення з управління мережею, сумісного з типом мережі та її призначенням, є важливим завданням. Зростання питань, пов'язаних з управлінням, призвело до ідеї впровадження мереж з автономними можливостями. У 2001 році IBM представила автономні системи. Мета полягала в тому, щоб досягти самонастроюваних, самовідновлювальних, самозахисних, самооптимізуючих і самокерованих систем. Сутності в автономній системі повинні бути здатні адаптуватися до свого оточення і самостійно приймати відповідні рішення. Ця адаптація є результатом постійного протікання зв'язку між сутностями системи.

Автономна мережа полегшує взаємозв'язок систем і успадковує поведінку від її конструктивних елементів. На сьогоднішній день впроваджені різні адаптації автономних мереж. Різні компанії надають різні моделі проектування автономних мереж, виходячи зі своїх потреб. Запропонована модель проектування автономних мереж ввела ієрархію шарів для спільного використання завдань управління мережею між різними сутностями. В управлінні мережею буде відігравати важливу роль розроблена топологія, яка буде надавати інформацію

про оточення вузлів. Процес збирає інформацію про сусідів вузлів всередині домену і створює карту кожного зв'язку між будь-якими двома сутностями всередині домену. Основними споживачами карти топології є сервіси з маршрутизації, нав'язування політики і т.д. Топологія допомагає вузлам краще зрозуміти своїх сусідів і оточення. У більшості випадків мережа не статична і змінюється з часом, наприклад, в спеціальних бездротових сенсорних мережах. Оновлення карти топології може бути складним завданням, враховуючи волатильність деяких мереж.

Технічне обслуговування топології - це процес, за допомогою якого оновлюємо карту топології та відображаємо оновлення. Зі зростаючою інфраструктурою результат процесів може допомогти підвищити продуктивність вузлів в різних ситуаціях. Однак самі процеси також споживають значну пропускну здатність. Були запропоновані рішення в різних типах мереж для зниження вартості споживання пропускної здатності. Ефективність процесів сильно залежить від структури мережі.

Чітко визначений підхід з високою ефективністю в лінійній мережі може постраждати від низької продуктивності в деревоподібної мережі або повністю підключеної мережі. Дати загальне рішення, придатне для всіх типів мереж, є складним завданням.

Впроваджена модель проектування автономних мереж залежить від сусідніх вузлів, які постійно спілкуються між собою. Така топологія відповідає топології «сніжинка», а сама системиа може розглядатись як мультикомп’ютерна система. Крім того, така система виступає як автономна система.

Отже, створення автономної системи на базі мультикомп’ютерних систем з використанням топології «сніжинка» дає змогу досягти більш високої продуктивності, зниження витрат і досягнення цілей користувачів є дуже актуальним зараз, коли функціонують мультикомп’ютерні системи і потрібно встановити ефективний зв’язок між їх вузлами.

Метою роботи є розробка мультикомп’ютерні системи на базі топології «сніжинка».

Об’єктом дослідження є процеси взаємодії між вузлами кластерів.

Предметом дослідження є мультикомп’ютерні системи та протоколи взаємодії і обміну повідомленнями між їх вузлами.

1 ПОНЯТТЯ АВТОНОМНИХ МЕРЕЖ

1.1 Архітектура автономних мереж

Автономні мережі були введені як нова схема управління мережею для боротьби з конфігурацією, захистом та відновленням у швидко зростаючій мережевій інфраструктурі. В роботі [1] припускають, що сутності автономної мережі можна розглядати як компоненти plug-and-play до мережі. Це означає, що вузли повинні бути здатні організовуватися з мінімальним наглядом і адмініструванням з боку зовнішніх елементів. Автономні мережі фокусуються на мінімізації залежності від адміністраторів-людей або інших центральних утворень [1]. Однак мета тут полягає не в тому, щоб повністю усунути людину-адміністратора або центрального керуючого суб'єкта [2]. Людський елемент і центральний суб'єкт необхідні для таких завдань, як прийняття рішення або нав'язування політики і нагляду [7].

Ієрархія всередині автономних вузлів дозволяє їм мати загальну інфраструктуру в групі автономних вузлів для спільного використання серійних вад, що надаються на найнижчому рівні. Автономні вузли мають свої цикли управління, використовуючи послуги, що надаються на більш високих рівнях ієрархії. Ця спільна інфраструктура забезпечує зв'язок між вузлами та інформацію для локальних функціональних можливостей вузла. Середній рівень дозволяє отримати доступ до спільної інфраструктури, виступаючи в якості контролера для вузлів. Загальна інфраструктура надає різні послуги вузлам. Середній шар - це сукупність всіх послуг, що надаються спільною інфраструктурою. Середній шар отримує інформацію із загального ущільнення і подає її у вищі шари [3]. Вищий рівень складається з атомарних сутностей і функцій, які дозволяють автономному вузлу використовувати послуги, які надають нижні шари. Запропонована архітектура складається з наступних трьох шарів: автономна мережева інфраструктура (AMI); автономна площина управління (АПУ); агент автономного

обслуговування (ААО). На рис. 1.1 зображена архітектура автономної мережі з точки зору вузла та проілюстровані різні її шари.

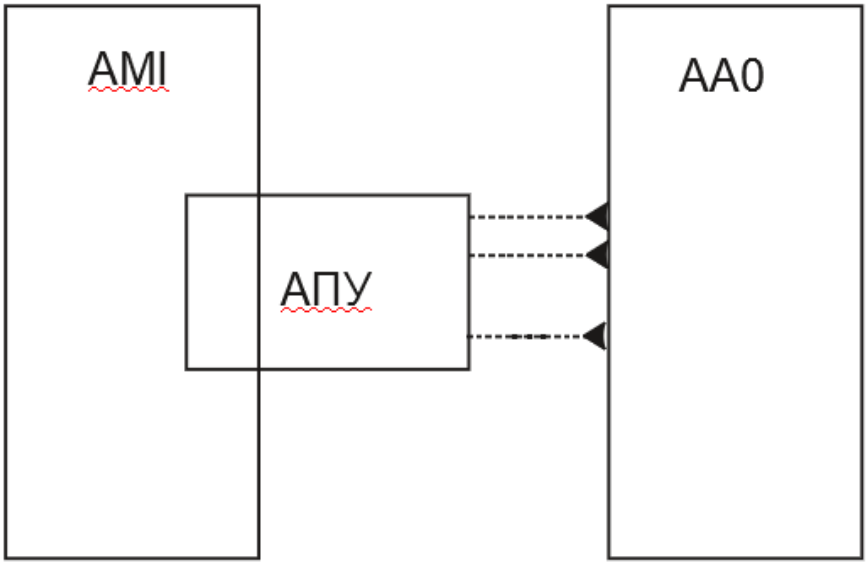


Рисунок 1.1 - Архітектура автономних вузлів

AMI забезпечує спільну інфраструктуру між вузлами. Існує безліч вже існуючих автономних функцій. У них є власні протоколи для спілкування, виявлення послуг тощо. AMI забезпечує спільну однорідну інфраструктуру для всіх автономних функцій для синхронізації значень, узгодження параметрів, виявлення послуг тощо [1]. Спільний набір можливостей для автономних функцій всередині автономного домену називається AMI. AMI, також, відповідає за зберігання як вузлової, так і загальної інформації. AMI надає необхідну інформацію для автономних функцій, таких як іменування та адресація кожного автономного вузла або полегшення зв'язку між вузлами.

АПУ є самонастроюваною комунікаційною інфраструктурою, яка в першу чергу служить площиною управління автономними функціями [1]. АПУ - це вузлова сутність, яка забезпечує шлях між AMI та автономними функціями, які намагаються використовувати послуги від AMI. АПУ - це автоматично побудована комунікаційна інфраструктура, яка є безпечною, стійкою та придатною для повторного використання [4]. АПУ виступає в якості *віртуального*

позасмугового каналу для вегетативних функцій [4]. Зв'язок АПУ здійснюється за принципом «перехід за стрибком», тобто інформація та повідомлення передаються від вузла до сусідів, поки він не досягне місця призначення. Тому, багато взаємодій між вузлами і сервісами спираються на таблицю суміжності, надану AMI [1, 4]. AMI несе відповідальність за забезпечення інфраструктури, необхідної для таких дій, як зв'язок та виявлення послуг, а АПУ надаватиме ці послуги вищому рівню та автономним функціям.

Використовуючи можливості AMI, ми можемо виробляти вегетативні функції. Автономні функції формуються з атомних сутностей, які ми називаємо ААО [6]. АСК встановлюються окремо напередодні автономного вузла. Існує зв'язок «один-до-багатьох» між автономним вузлом і ААО, що означає, що кожен автономний вузол може встановити кілька ААО.

Компоненти разом утворюють автономні мережі. У зв'язку з деякими функціональними особливостями автономної мережі, перевага віддається IPv6 перед IPv4 [5]. IPv6 підтримує деяку автономну поведінку, таку як самостійне налаштування адрес на фізичних інтерфейсах [3]. Автономна мережа високого рівня зображена архітектурою на рис. 1.2.

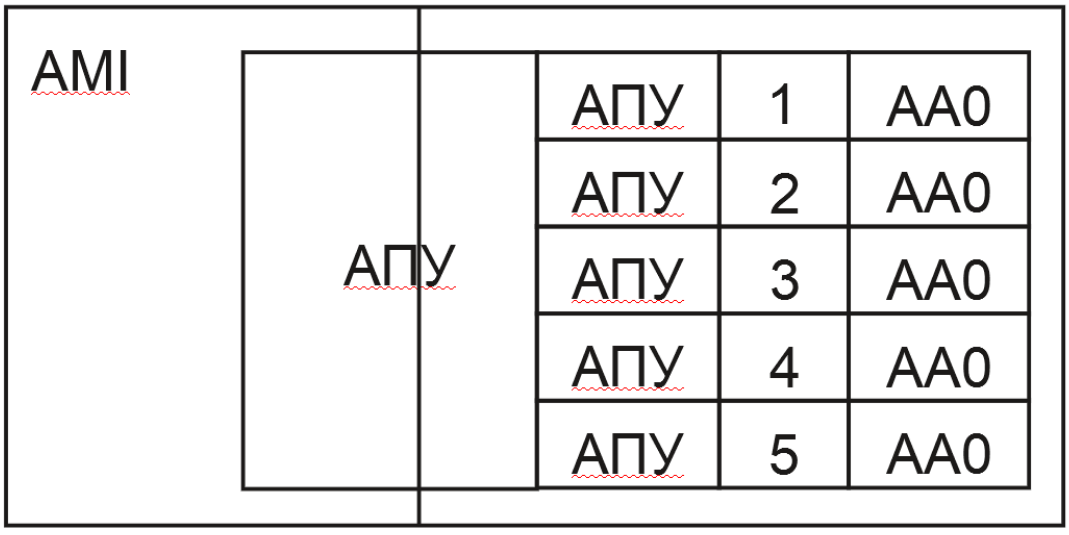


Рисунок 1.2 - Автономна мережа високого рівня

Розглянемо саме IPv6, його області адрес і обгрунтування його переваг при виборі інтернет-протоколів для автономних мереж. IPv6 був представлений як наступник IPv4 з більшим простором адресації, намагаючись вирішити проблему вичерпання IPv4-адреси. IPv4 може містити 32-бітну адресацію, тоді як IPv6 здатний містити 128-бітні ідентифікатори адрес. Розглянемо дві основні області адресації в IPv6: адреса Link-Local (LL) та унікальна локальна адреса (ULA). Адреси LL не є маршрутизованими, але можуть генеруватися вузлом автономно. IPv6-адреса представлена в 8 групах 16-бітних адрес, розділених двокрапками. Кожна 16-розрядна група відображається чотирма шістнадцятковими розрядами. Використання LL-адрес може призвести до менших таблиць маршрутизації та зниження ризику атак за рахунок зменшення впливу мережі на шкідливі вузли. З іншого боку, ULA є глобально унікальною адресою, призначеною для зв'язку всередині домену і має локальну маршрутизацію [6, 7].

Конфігурація адрес IPv6 LL є автономною. Адреса LL може бути згенерована автоматично за допомогою ідентифікатора інтерфейсу. Префікс адреси LL фіксований, *FE80::/10*, а наступні 54 біти - 0. Останні 64 біта адреси - це ідентифікатор інтерфейсу, який генерується, використовуючи MAC-адресу (з використанням формату EU I-64), або це може бути 64 випадкових біта.

ULA є глобально унікальною, але використовується для локальних мережевих цілей. Унікальність дає йому ту перевагу, що навіть при витоку за межі локальної мережі він все одно не викликає конфліктів з іншими адресними просторами або доменами [8]. Якщо мережі об'єднуються або розширюються, оскільки адреса є глобально унікальною, то відсутні все-таки будь-які конфлікти в адресах. ULA має префікс *FC00::/7*, за яким слідує 1-бітовий локальний/глобальний ідентифікатор, 40-бітовий глобальний ідентифікатор, 16-бітний ідентифікатор підмережі та 64-бітний ідентифікатор інтерфейсу. Наявність певної приставки дозволяє проводити швидку фільтрацію [9]. RFC4861 [10] вводить протокол Neighbor Discovery (ND), який дозволяє вузлам виявляти своїх сусідів шляхом визначення їх адреси. Протокол ND також має вегетативну поведінку. Протокол ND не тільки повертає список сусідів. Він служить більше.

Завдяки своїй автономній поведінці, протокол ND забезпечує рішення наступних властивостей мережі [11]: виявлення префіксів, виявлення параметрів, виявлення маршрутизатора, автоконфігурація адреси, роздільна здатність адреси, визначення наступного переходу, дублювання адреси detection, і redirect. Виявлення префікса дозволяє вузлам знаходити доступні місця призначення мережі на кожному посиланні. Виявлення параметрів дозволяє вузлу з'ясувати специфікації зв'язку, такі як максимальний блок передачі. Конфігурація адрес дозволяє вузлам задавати адреси інтерфейсів бездержавним способом [12]. Визначення наступного стрибка дозволяє вузлам розміщувати трафік на сусідній ланці до кінцевої долі. Виявлення повторюваних адрес дозволяє вузлам перевірити, чи адреса, яку вони хочуть використовувати, вже використовується, щоб запобігти зіткненню. Протокол ND, також, вводить п'ять типів ICMP-повідомлень, які полегшують роботу вищезгаданих служб. Повідомлення ICMP такі: запит маршрутизатора, реклама маршрутизатора, запит сусіда, реклама сусіда та перенаправлення.

Повідомлення про запит маршрутизатора дозволяє вузлам змусити маршрутизатори рекламувати свою присутність перед запланованою рекламою. Маршрутизатори з підтримкою IPv6 потім надсилають повідомлення про рекламу маршрутизатора після отримання запиту маршрутизатора або досягнення запланованого часу. Рекламні повідомлення маршрутизатора включають більше, ніж адресу маршрутизатора. Вони надають вузли з інформацією, такою як префікси, параметри конфігурації адрес, hop-limit тощо. Повідомлення ICMP про запит сусіда використовуються для пошуку адреси LL сусіда або перевірки того, чи активні вони ще чи ні. За запитом сусіда слідує відповідь на рекламу сусіда. Вузол, також, може розсилати сусідські оголошення, не отримуючи клопотання сусіда, щоб повідомити про свою присутність сусідам.

В рамках своєї автономної поведінки маршрутизатори можуть інформувати вузли про кращий наступний перехід, відправляючи повідомлення перенаправлення ICMP на вузли. Згідно з визначеннями, деякі повідомлення ICMP, такі як реклама перенаправлення або маршрутизатора, призначені для

використання маршрутизаторами, а не хостами. АПУ налаштовується на основі інформації, яку вона отримує від своїх сусідів. Використовуючи LL-адресу, зв'язок між сусідами відбувається без проходження таблиці маршрутизації. AMI забезпечує зв'язок між вузлами. Однорангові вузли можуть бути несусідніми вузлами в домені. Тому, запропоновано використовувати як адреси LL, так і ULA для комунікаційних цілей, таких як переговори та синхронізація.

1.2 Протоколи взаємодії

AMI несе відповідальність за надання спільної платформи для комунікаційних послуг, таких як переговори, синхронізація та виявлення. Протокол автономної сигналізації GenRic розроблений, щоб забезпечити легке вираження цих функцій. Зв'язок між сутностями автономної області, починаючи з ранніх стадій завантаження до завершальних кроків завершення будь-якого каналу зв'язку, полегшується GRASP. GRASP - це розширений протокол, який забезпечує зв'язок між різними автономними функціями та вегетативними вузлами.

Різні протоколи та механізми в автономних мережах, включаючи запропоновані методи та реалізації, використовують GRASP, щоб автономні вузли могли спілкуватися. GRASP - це протокол зв'язку, підписаний для надання API для вираження архітектурних концепцій відповідно до бачення автономної мережі NMRG. GRASP представив нову структуру даних, здатну позначати та приймати значення. Ця структура даних називається *ціллю GRASP*. Кожна мета GRASP буде відображена в AAO. GRASP дозволяє нам реєструвати AAO та використовувати ім'я AAO як унікальний ідентифікатор для майбутніх посилань. Для того, щоб надати інформацію, необхідну для автономних функцій, GRASP представив структуру даних, здатну містити різні ідентифікатори та значення. Ім'я використовується як унікальний ідентифікатор цілей GRASP, а значення містить фактичне значення цілей GRASP. *Значення* може містити будь-який тип даних, що дозволяє GRASP підтримувати обмін будь-якою структурою даних у домені. Поряд з *назвою* та *значенням,* є кілька прапорців, що супроводжують кожну мету

GRASP. *Об'єктивними FLAGS* є Neg (встановлено значення True, якщо ціль GRASP підтримує переговори), с*инхронізація (*встановіть значення True, якщо ціль GRASP підтримує синхронізацію), с*ухий (*встановлюється в true, якщо ціль GRASP підтримує переговори про сухий хід). Іншим важливим полем у структурі даних цілей GRASP є *кількість циклів*, яка вказує на максимальну кількість кроків для цілей переговорів.

GRASP обмежує *ім'я* цілей GRASP форматом *рядка*. Кількість *циклів* може містити будь-яке значення (*ціле число*) від 0 до 255. Всі цілі GRASP, незалежно від їх значення, повинні передаватися через домен між вузлами в одному байтовому форматі. GRASP використовує *CBOR,* формат двійкової серіалізації даних [12]. Перш ніж ініціатор повідомлення надішле цілі GRASP, він перетворить значення цілі GRASP на об'єкт CBOR, а потім передасть його вузлам. На приймаючій стороні приймач перетворить *об'єкт CBOR* в базовий формат *цілі GRASP*.

Кожна мета GRASP повинна бути зареєстрована AAO. GRASP вводить іншу структуру даних, яка називається *позначеною ціллю*. Тегований об'єкт містить лише два значення: зареєстровану ціль та вказівник на об'єкт AAO, на якому зареєстрована ціль GRASP. Позначена мета - це фактична структура даних, якою обмінюються всередині автономного домену. GRASP призначає випадковий номер порту кожній цілі, зареєстрованій на AAO. Поки автономний вузол працює і AAO зареєстрований на цьому автономному вузлі, номер порту не буде порушений.

GRASP підтримує безліч повідомлень для різних цілей. Кожен з них має певний формат разом із певним набором прапорів. Обмін повідомленнями здійснюється з метою виявлення, узгодження або синхронізації. Кожне повідомлення GRASP має ідентифікатор типу повідомлення. Доступні типи повідомлень: повідомлення про надання відомостей (повідомлення із запитом на надання відомостей; повідомлення про відповідь на виявлення); повідомлення про переговори (запит на переговори; переговори; підтвердити очікування; завершити переговори); синхронізації (запит на синхронізацію; синхронізації;

синхронізація повідомлень). Кожне повідомлення містить ідентифікатор сеансу. Ідентифікатор сеансу генерується випадковим чином ініціатором з'єднання, будь то виявлення, переговори або синхронізація, і приймаючий вузол повинен використовувати той самий ідентифікатор сеансу для будь-яких подальших повідомлень, що стосуються цього зв'язку. Ідентифікатор сеансу генерується випадковим чином і двічі перевіряється GRASP, щоб уникнути будь-якого зіткнення.

Процес відкриття дозволяє автономним вузлам знаходити тих, які зареєстрували конкретну мету GRASP на конкретному AAO. В результаті процесу відкриття, функція виявлення GRASP поверне один або кілька локаторів, які містять значення адреси ULA кореспондуючих автономних вузлів і номер порту, на якому доступний доступ. Дисковий вузол слухає вхідні запити для тієї ж мети GRASP (з тією ж назвою) і зареєстрований в AAO з тією ж назвою.

Для того, щоб відкриття GRASP знайшло одноранговий вузол (або кілька вузлів), назва цілі GRASP та її відповідний AAO від вузла повинні відповідати імені від ініціатора відкриття GRASP. Локатор, також, містить значення номера порту, який був присвоєний меті, яку шукаємо на відповідному автономному вузлі. Оскільки номер порту випадковим чином присвоюється кожній цілі на кожному вузлі, його не можна вгадати або зібрати будь-якими іншими способами, крім запиту безпосередньо з вузла-вузла. Запит на виявлення GRASP надсилається на всіх інтерфейсах. Локатор — це об'єкт класу як локатор, який містить значення адреси ULA або LL вузла, інтерфейс, на якому автономічний вузол повинен достукатися до вузла, і екземпляр цілі GRASP. Мета GRASP, також, міститиме номер порту, за яким він може бути. Процес відкриття використовується для пошуку аналогів і часто супроводжується переговорами про значення мети. Тому, для цілей відкриття цілі повинні мати прапор *NEG* встановлений на true. Відкриття GRASP - це процес переходу за стрибком, що означає, що запит буде передаватися з одного вузла на інший. На цьому шляху, якщо який-небудь вузол задовольняє запит, відповідь буде відправлена назад по тому ж шляху і набору вузлів, які запит взяв, щоб досягти відповідного вузла. На зворотному шляху до ініціатора всі вузли оновлять свій кеш за допомогою локатора відповідного вузла.

Процес виявлення GRASP починається з надсилання повідомлення про виявлення від ініціатора на всіх інтерфейсах. На першому кроці всі сусіди отримають повідомлення із запитом на відкриття. Тепер може статися один з трьох наступних результатів. Автономний вузол, який отримує запит на виявлення утримує запитану ціль і надішле повідомлення з відповіддю на виявлення. Що стосується другої можливості, то одержувач не є власником мети і не володіє будь-якою інформацією про інших власників, яка задовольняє запит про виявлення. Таким чином, запит буде переданий сусідам, за винятком інтерфейсу, на якому запит був отриманий. Що стосується третьої можливості, сусід, який отримав запит на відкриття, раніше кешував локатори сусідів, які відповідають запитуваній меті. У цьому випадку, якщо кешована інформація ще не закінчилася, то буде сформовано повідомлення-відповідь, що містить локатори з кешованої інформації. Функція відкриття поверне список (розміром один або більший), який містить локатори вузлів, які мають цю мету відкриття. Пізніше локатори будуть використовуватися для ініціювання переговорів з колегами. Зонди відкриття будуть поширюватися в домені, поки вони не знайдуть відповідний аналог або поки термін їх дії не закінчиться. Щоб запобігти нескінченному пошуку у великій області, де зонди безперервно заглиблюються в область для пошуку мети, функція виявлення GRASP ввела *тайм-аут* як параметр зонда відкриття. Виявлення GRASP - це односторонній запит. Це означає, що, якщо вузол виявить вузол, запустивши GRASP, не гарантується, що вони знайдуться. Результат процесу виявлення GRASP буде використаний для цілей переговорів. Для переговорів GRASP запит на переговори повинен бути надісланий від ініціатора переговорів до відповідного колеги. Після цього запиту буде відповідь на переговори або повідомлення про завершення переговорів від колеги. Якщо колега-відповідач приймає пропозицію, переговори закінчуються повідомленням при прийом у повідомленні кінець пошуку. В іншому випадку їхні колеги можуть продовжувати переговори щодо цінності цілі, обмінюючись повідомленнями про переговори. Якщо вузол, що відповів, не приймає запропоноване значення, повідомлення про відхилення відповіді буде надіслано в повідомленні кінець пошуку.

Під час переговорного процесу будь-яка сторона може завершити переговори, прийнявши або відхиливши запропоновані цінності. Логіка переговорів може змінюватися в залежності від мети вегетативної функції. Тому, адміністратор мережі несе відповідальність за встановлення логіки переговорів. Обидва колеги продовжуватимуть переговори до тих пір, поки не досягнуть угоди або не буде досягнута максимальна кількість *переговорних кроків*. Після переговорів ніякі інші повідомлення, такі як підтвердження припинення, не будуть передані.

Переговори GRASP приймають позначену мету, локатор цільового сусіда, як вхідні дані. Локатор отримується з процесу виявлення GRASP, або ініціатор буде використовувати кешовану інформацію для отримання локатора вузла. Для зазначеної мети прапорець *NEG* повинен бути встановлений на true. Переговори GRASP приймають локатор відповідного одноранга як параметр. Зазвичай переговори GRASP відбуваються після виявлення GRASP для отримання локатора необхідного вузла. Але, якщо припустимо, що якимось чином можна зібрати номер порту та адресу ULA автономного вузла, на якому встановлена мета GRASP, то можна створити локатор і встановити переговори. Якщо локатор не надається як параметр для переговорів GRASP, відкриття буде запущено першим, і буде обраний перший вузол у списку знайдених вузлів.

Узгодження і синхронізація - це два основних способи комунікації, введені в [13]. Під час переговорів основна увага приділяється досягненню угоди між двома вузлами, процес синхронізації не супроводжується кількома повідомленнями, що обговорюють мету. Існує два типи синхронізації: «одноадресна синхронізація» і «багатоадресна синхронізація». У разі одноадресної синхронізації ініціатор синхронізації розмикає з'єднання з відповідачем синхронізації (ініціатор вже знає локатор відповідача) і відправляє повідомлення і отримує у відповідь.

Подальший обмін повідомленнями не здійснюватиметься. У разі одноадресної синхронізації, якщо ініціатор синхронізації не знає локатора вузла, буде використовуватися режим. У цьому випадку ініціатор надсилає повідомлення

GRASP, що містить ціль синхронізації. Це дозволяє повідомленню виявлення діяти як запит синхронізації. Цей тип запиту, дозволяє відповідачу виявлення відповісти на синхронізаційне повідомлення замість відповіді ініціатору запиту виявлення [14]. Коли велика група вузлів бажає синхронізувати значення однієї і тієї ж мети, то використовуємо GRASP flood і синхронізацію. Ініціатор повідомлень надішле небажане повідомлення синхронізації повені на всіх інтерфейсах.

Повідомлення про потік повідомлень - це багатоадресне повідомлення всім сусідам. Одержувач потоку повідомлень передасть це повідомлення своєму сусіду(крім того, що приймається). Щоб запобігти нескінченному затопленню, GRASP додає кількість циклів. Отримавши повідомлення про потік повідомлень, приймач зменшить значення кількості циклів на одиницю, а потім передасть його іншим сусідам. Будь-який автономний вузол буде використовувати для цієї мети багатоадресну передачу ЛЛ [15]. Багатоадресна передача LL прив'язана лише до одного стрибка, але для цілей затоплення потрібно використовувати багатоадресну передачу LL. Вузли *не* будуть ретранслювати точно такі ж багатоадресні повідомлення LL на інші вузли. Кожен вузол змінює початкову багатоадресну передачу LL, в цьому випадку змінюючи кількість циклів. Отже, повідомлення не є однаковим і було змінено. З міркувань безпеки GRASP представляє функцію для зв'язку в незахищених середовищах. Деякі дії, такі як пошук сусідніх АПУ через проксі-сервер приєднання, є небезпечними, оскільки вузол приєднання може бути шкідливим. Щоб запобігти цьому, функція GRASP обмежує радіус зв'язку лише одним переходом у DULL. Це означає, що при використанні DULL автономні вузли будуть спілкуватися тільки з сусідами, а будь-які повідомлення в DULL не будуть передаватися іншим сусідам. Ініціатор DULL може надсилати повідомлення про виявлення або синхронізацію потоку повідомлень. Одержувач цих запитів перевірить кількість циклів повідомлень, і якщо вони більші за одиницю, повідомлення від ініціатора буде відкинуто. Приймачr також не буде ретранслювати будь-які багатоадресні повідомлення LL, оскільки вони прив'язані лише до зв'язку між сусідами. Автономні вузли можна розглядати як пристрої

Plug-and-Play [15-17]. Оскільки самозабезпечення є однією з характеристик автономних мереж, кожен вузол повинен мати можливість знайти спосіб безпечного зв'язку та автономного використання свого АПУ. Під час завантаження задіяно кілька автономних вузлів з різними ролями, тобто реєстратор, проксі-сервер приєднання. Він дозволяє вузлам досягти стану взаємної аутентифікації, обмінюючись сертифікатами. Він видає сертифікати з організаціями для приєднання до мережі для авторизації завантаження АПУ та безпечного зв'язку з сусідами.

Сертифікат - це формат сертифіката відкритого ключа, який в основному використовується в захищених транспортних протоколах, наприклад, SSL. Основну увага приділимо потоку повідомлень між сутностями домену при приєднанні, а не стандартним засобам забезпечення безпеки.

Після приєднання до домену новий вузол не має повністю працездатного АПУ. Новий вузол, який ще не зареєстрований всередині домену, називається заставою. Кожен домен має центральну організацію, яка називається реєстратором, відповідальним за аутентифікацію будь-якої застави, яка бажає приєднатися до домену.

Реєстратор приймає або відхиляє прохання заставодавця про приєднання до домену. Застава повинна зв'язатися з реєстратором, щоб отримати автентифікацію, але він не знає домену і, тому, не знає локатора реєстратора, щоб зв'язатися з ним. Крім того, виробник вбудовує в пристрій застави сертифікат, який дозволяє аутентифікувати пристрій. Застава безпосередньо не зв'язується. Реєстратор несе відповідальність за прохання до нього автентифікувати пристрій та його власника [18-23]. Таким чином, він відіграє важливу роль у досягненні заставою та реєстратором взаємної аутентифікації. Для того, щоб отримати автентичність, застава повинна надіслати реєстратору запит на зв’язок. Застава не може використовувати метод виявлення GRASP для визначення місцезнаходження реєстратора через заходи безпеки. Він може контактувати лише зі своїми безпосередньо підключеними сусідами на рівні каналу зв'язку і лише через адресу LL, перш ніж реєстратор автентифікує його [24-31]. На цьому

етапі, щоб зменшити ризики безпеки, зв'язок між заставою та її сусідами, які є аутентифікованими учасниками мережі, обмежена [31-36]. З міркувань безпеки реєстратор не може заповнювати свою присутність, оскільки викриття себе може бути ризиком безпеки [37-41], що дозволяє зловмисним вузлам зловживати інформацією. Проксі-сервер приєднання - це середній вузол, що з'єднує заставу з реєстратором, який вже є автентичною частиною домену. Застава знає, що для того, щоб потрапити до реєстратора, він повинен спочатку пройти через приєднаний проксі. Отже, він буде шукати проксі-сервер приєднання серед своїх сусідів. Знайшовши проксі-сервер приєднання, застава надішле повідомлення із запитом, призначене реєстратору та передано через проксі-сервер приєднання. Якщо буде виявлено кілька проксі-серверів для приєднання, буде використано перший виявлений. Структура ваучерного запиту, що містить інформацію про пристрій застави, наприклад «серійний номер». Коли застава отримає підтверджуючий запит ваучера, то вона буде діяти як проксі приєднання [35]. Проксі-сервер приєднання перешле реєстратору ваучер запиту на заставу. Реєстратор запустить набір попередніх операцій аутентифікації запиту і перешле його після додавання додаткової інформації про себе з метою аутентифікації. Після успішної аутентифікації пристрою застави та реєстратора, він повідомить як заставу, так і реєстратора в одному повідомленні. Частина, призначена для пристрою застави, буде зашифрована за допомогою тих самих ключів, що він вбудував у заставу. Якщо запит недійсний, це означає, що пристрій застави або реєстратор є шахрайськими або інформація надсилається до нього неповною, повідомлення про те, що запит недійсний, буде відправлено назад реєстратору та заставу. Якщо реєстратор бажає прийняти заставу, то він надішле відповідь на ваучер через той самий проксі-сервер приєднання. Застава отримає відповідь, і якщо вона дійсна, то застава приєднається до домену та налаштує його АПУ. Відтепер застава є частиною домену і може безпечно зв'язуватися з іншими суб'єктами за допомогою сертифіката, виданого реєстратором.

1.3 Постановка задачі

Тема створення мультикомп’ютерні системи на базі топології «сніжинка» є актуальним завданням.

Недостатність стандарту IPv4 спонукатиме перехід до IPv6. Наявність багатьох вузлів в комп’ютерних мережах потребує узгодження їх взаємодії через відповідні протоколи та засоби взаємодії. Все це може бути вирішене створенням мультикомп’ютерних систем для розв’язання певного класу задач.

Вирішення поставленого завдання передбачає виконання наступних етапів:

1) аналіз відомих систем та засобів, що реалізують стандарт IPv6;

складання вимог до топології;

2) проєктування структури системи та протоколів для отримання топології;

3) розробка рішення з використанням централізованого і децентралізованого підходів до отримання оптимальної топології;

4) реалізація запропонованого рішення та проведення експерименту з аналізом отриманих результатів.

2 ПРОЄКТУВАННЯ ТОПОЛОГІЇ

2.1 Вибір топології

Відкриття топології — це процес знаходження зв'язків між кожною парою сутностей у мережі. Результат цього процесу може дозволити автономним вузлам краще зрозуміти своє оточення і зробити такі процеси, як маршрутизація або відновлення після збоїв, швидшими та ефективнішими. Інший варіант використання результатів - це програмно-визначена мережа (ПВМ). ПВМ - це новий підхід до управління центральною мережею. ПВМ пропонує відокремити площину управління від площини даних для підвищення досвіду управління мережею [39]. Керуючи програмним забезпеченням мережевих пристроїв, наприклад, комутаторами або маршрутизаторами з підтримкою ПВМ, управління та моніторинг мережевих сутностей буде відбуватися набагато легше через центральний контролер ПВМ. В роботі [19] пропонується переваги знання топології мережі для мульти кастингу в ПВМ. Для класичної багатоадресної передачі напрямки невідомі. Окремі маршрутизатори знають лише про те, що до одного з його інтерфейсів підключено один або кілька зацікавлених приймачів. По суті, розробляється набір найкоротших шляхів, від джерела до всього набору пунктів призначення. В контексті ПВМ, централізований контролер може мати уявлення про загальну топологію, і, тому, може налаштувати перемикачі для забезпечення різних оптимізацій. Таким чином, робота ПВМ в цілому залежить від того, що центральний контролер знає фактичну топологію мережі, а багатоадресний ПВМ може використовувати інформацію топології для проведення залежної від додатків оптимізації. Таку топологію можна класифікувати за трьома основними шарами. Накладання топологій на рівні мережі (на рівні Інтернету) та на рівні каналу. Мережевий рівень фокусується на топології вищого рівня в міжмережевій мережі. Мережа - це не тільки фізичний зв'язок між двома кінцевими вузлами в одному домені. Сьогоднішній Інтернет набагато більший і призначений для підключення різних мереж. Маршрутизація може здійснюватися всередині домену між двома

сутностями в одній підмережі за допомогою відомих протоколів, таких як протокол маршрутизації стану зв'язку або протокол дистанційно-векторної маршрутизації, або це може бути виконано між двома незалежно працюючими мережами з різними адресами і підмережами за допомогою відповідних протоколів. У разі маршрутизації стану зв'язку протоколи та вузли повністю знають топологію мережі і використовують цю інформацію для прийняття рішення про найкоротший шлях. Що стосується дистанційно-векторних протоколів, вузли матимуть часткову інформацію про доступну їм топологію. Топологія накладання фокусується на P2P-з'єднанні системи, а топологія на рівні каналу зв'язку фокусується на фізичному зв'язку між двома вузлами. Топологія канального рівня є основою багатьох завдань управління мережею. Топологія на рівні каналу зв'язку описує, як підключені пристрої в мережі. Топологія на рівні зв'язку забезпечить усі зв'язки між двома сутностями.

Постійний зв'язок відбувається для різних цілей, таких як покращення маршрутизації або оновлення/обмін параметрами конфігурації. Оскільки фізичний зв'язок між вегетативними вузлами має вирішальне значення, методології, які досліджують фізичний зв'язок між вузлами, більш вивчені. Інший спосіб класифікувати топологію базується на тому, як метод топологій збирає, зберігає та поширює інформацію. Всі згадані дії можуть бути виконані централізовано або децентралізовано. У централізованому підході вся інформація збирається вузлом і зберігається на одному вузлі. Роль центрального органу управління визначається адміністратором мережі або вирішується виборчим процесом. У розподілених обчисленнях вибори лідера - це процес, в якому всі вузли беруть участь у виборі одного вузла в якості свого лідера. Всі вони надсилають дані або спілкуються з цим єдиним обраним вузлом замість того, щоб передавати їх на всі вузли. Обраним вузлом-лідером може бути розподіл даних і відповідь на запити або полегшення комунікації між двома вузлами [34]. Евристика залежить від випадкових факторів або специфічної для вузла інформації, такої як ідентифікатор вузла (ціле число) в мережі. Наприклад, вузол з найбільшим значенням ідентифікатора може бути обраний в якості лідера. Однак

ефективність виборчого процесу корелює з розміром чистої роботи. Зі збільшенням розміру мережі зростає і час і кількість повідомлень, необхідних для цього процесу. Другий підхід є розподіленим підходом, де різні вузли беруть участь у передачі інформації топології від інших вузлів. Інформація про процеси може зберігатися розподіленим способом між декількома вузлами. Зберігання карти топології може бути різним в розподілених системах. Всі вузли, відповідальні за утримання карти топології, можуть мати репліки карти топології. Цей тип зберігання даних називається реплікацією в термінах розподілених систем. Або кожен відповідальний вузол може містити частину карти топології. Агреговані збережені топологічні дані з усіх вузлів, які зберігають частину карти, дають повну топологічну карту мережі. Залежно від типу мережі можуть використовуватися різні методології. На різних стадіях топології можуть використовуватися різні методи. Наприклад, можна запропонувати новий метод для фази налаштування карти топології, і можна використовувати діагностичні інструменти для цілей топології.

Відкриття топології було досліджено в багатьох контекстах, і багато існуючих методологій залежать від традиційних інструментів і протоколів, таких як просте управління мережею.

SNMP використовується для управління мережею та управління пристроями шляхом збору інформації та її організації. ICMP використовується для діагностичних і моніторингових цілей в IP-мережах [37]. ARP призначений для відображення IP-адреси з фізичною адресою інтерфейсів, тобто MAC-адресою [23].

Передбачуваний мережевий контролер повинен мати повний список сутностей мережі. Саме тому він не є підходящим підходом для цієї топології. Контролер може використовувати повідомлення ICMP для перевірки того, чи активна сутність. Автономні вузли повинні мати можливість налаштовуватися при приєднанні до мережі без будь-якого попереднього знання мережевих сутностей і повинні отримувати налаштування параметрів тільки через зв'язок зі своїми аутентифікованими вузлами. Існують також проблеми безпеки

повідомлень ICMP, наприклад, блокування брандмауера та підробка ICMP-повідомлень. Тому, пропонується використовувати ICMP-повідомлення під час процесу класифікації топологій для мереж, які не є динамічними та мають стабільне з'єднання.

ARP використовується для вирішення IP-адреси до MAC-адреси інтерфейсів. Тим не менш, ARP не може знайти зв'язок між сутностями в мережі. Це може допомогти знайти існуючі сутності, але, він не може створити карту мережі, яка відображає зв'язок між сутностями. Також, в досить великих мережах ARP буде неефективний, так як не може представляти всі сутності через розмір таблиці ARP [34]. Однак, якщо потрібно, ARP можна використовувати для виявлення IP-адрес сусідів і зіставлення їх з відповідними MAC-адресами.

SNMP - це стандартний протокол IP, який використовується для цілей управління шляхом збору та організації інформації мережі. Для роботи SNMP обидва вузли (сервер SNMP і клієнт) повинні підтримувати SNMP, оскільки SNMP не завжди увімкнено з міркувань безпеки. Багато надлишкової інформації може бути передано, оскільки SNMP не був явно розроблений для цілей класифікації топологій. Також, SNMP вимагає попереднього знання мережі. Інші інструменти можуть бути використані для виявлення топології мережі, наприклад, traceroute. Як і ICMP, traceroute, також, є діагностичним інструментом в IP-мережі, який дозволяє знайти шлях і час туди і назад від джерела до пункту призначення. Як і попередні методи, traceroute також вимагає попереднього знання мережевих IP-адрес, і він не здатний представляти повне з'єднання між вузлами.

Автономна система — зв'язана група з одного або декількох наборів префіксів IP-адрес у віданні одного або декількох операторів Інтернет-мережі, яка має чітко визначені політики маршрутизації. Термін "префікс" є еквівалентом "CIDR-блок". За класичним визначенням автономна система представляє собою сукупність маршрутизаторів під управлінням єдиної служби технічного адміністрування, що використовує протокол внутрішнього шлюзу з чітко визначеними політиками маршрутизації IP-пакетів усередині себе, а за допомогою протоколу зовнішнього шлюзу маршрутизує IP-пакети в інші автономні системи.

Тому, автономні системи і мережі є ефективним рішенням в перспективі. Частковою реалізаєю його можу бути мультикомп’ютерна система з топологією «сніжинка» (рис. 2.1).

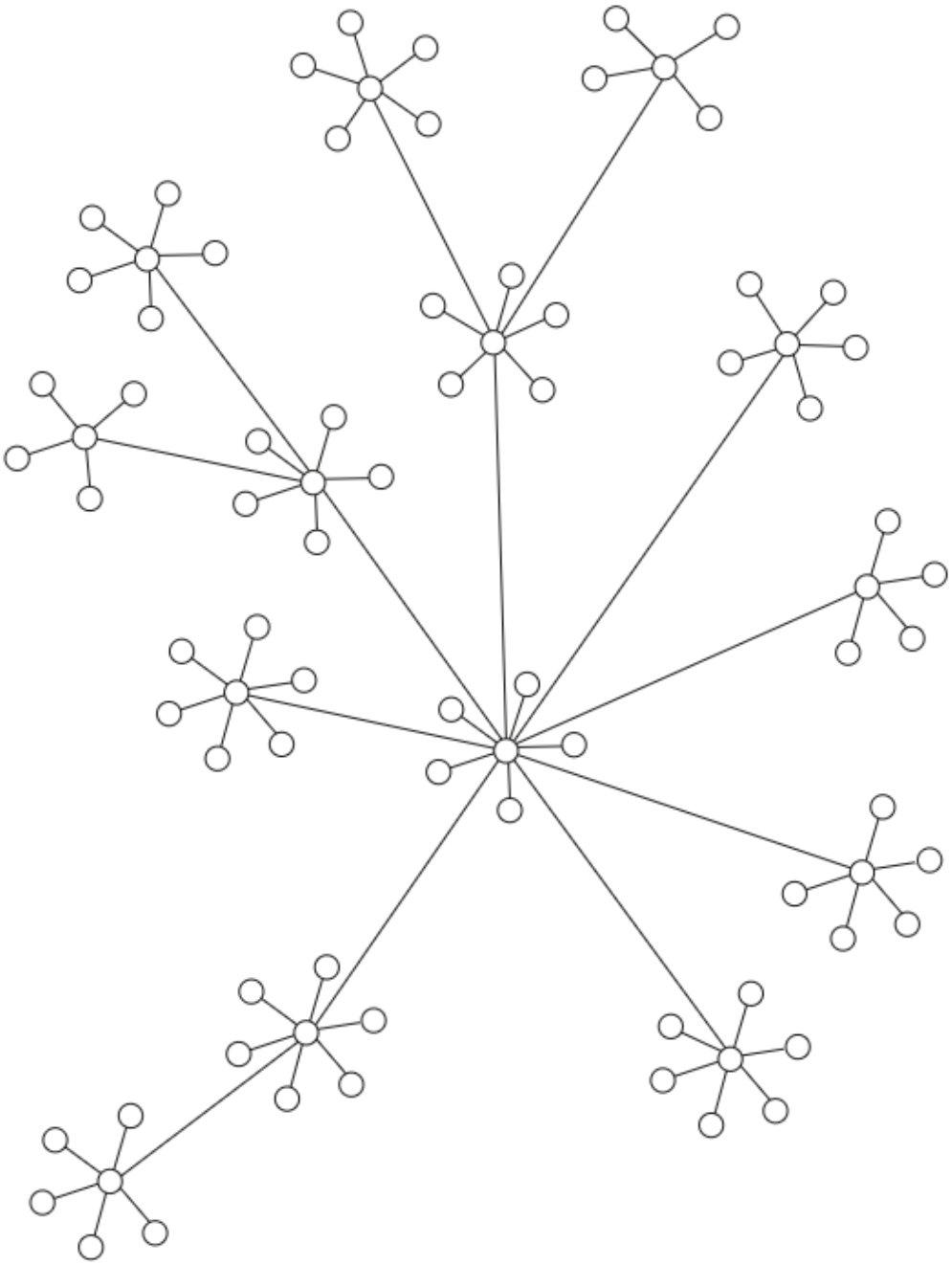


Рисунок 2.1 – Приклад топології «сніжинка»

2.2 Протокол виявлення мережі IPv6

Альтернативним підходом до пошуку топології мережі є збір інформації з протоколу. IPv6 дозволяє вузлам знаходити своїх сусідів і активно перевіряє їх доступність, відправляючи конкретні ICMP-повідомлення. Більшість комунікацій в протоколі обмежується зв'язком між сусідами шляхом відправки простих ICMP повідомлень. Тому, він не споживає значної кількості пропускної здатності. Однак інформація все ще локальна для вузла та його сусідів. Потрібен в такому випадку центральний вузол для збору цієї інформації та її обробки для генерації топології карти мережі. Кожен пакет бере протокол на первинному рівні протоколом зв'язку між контролером ПВМ та іншими сутностями. Він широко використовуваний методом для цілей класифікації топологій, але він має деякі обмеження. Він вразливий до багатьох атак, наприклад, фабрикації посилань, підміни перемикачів тощо. Пакети LLDP використовуються для інформування сусідів про інформацію вузла. Ця інформація може включати ідентифікатор протоколу, ідентифікатор порту тощо. Він періодично розсилається з кожного інтерфейсу.

При приєднанні до мережі нового комутатора контролер і комутатор почнуть обмінюватися повідомленнями. За цим повідомленням слідує запит функції від контролера. Відповідь функції включає стан портів, які налаштовані на зниження спочатку. Потім комутатори повідомлять контролеру про зміну стану своїх портів. Потім контролер почне надсилати пакети LLDP до *портів вгору* на комутаторах з певним шляхом, починаючи від одного комутатора і закінчуючи контролером, щоб визначити, чи є шлях між комутаторами. Після отримання пакетів LLDP контролер може генерувати карту топології. Щоб з'ясувати шляхи між будь-якими двома сутностями в роботі, повинен бути згенерований конкретний пакет. Для кожного шляху між будь-якою парою вузлів цей процес повинен відбуватися. Тому цей метод вкрай неефективний і трудомісткий. Крім того, це вимагає, щоб центральний вузол наказував іншим вузлам вживати заходів, що суперечить бажаній вегетативній поведінці. Цей метод категоризує вузли та порти

(інтерфейси) на кожному пристрої. Кожен вузол в цьому ПВМ може містити один з наступних станів: невиявлений вузол, листовий вузол, V-листовий вузол і вузол ядра. Метою контролера ПВМ є створення дерева управління, в якому кожен вузол знає свій батьківський вузол і шлях до контролера. Невиявлений вузол - це вузол, який ще не отримав ніяких запитів на приєднання до дерева управління. Листовий вузол - це зовнішній вузол, а V-листовий вузол має всі свої порти, з'єднані з іншими листовими або V-листовими вузлами. Решта вузли будуть розглядатися як основні вузли. Кожен порт має один із таких станів: резервний порт, батьківський порт, дочірній порт і обрізаний порт. Порт — це резервний порт, коли він не є частиною дерева керування. Порт вважається батьківським, якщо він є основним портом і спочатку отримав повідомлення на цьому вузлі. Порт — це дочірній порт, якщо він є нижнім портом і отримав повідомлення. Обрізаний порт - це дочірній порт, який отримав повідомлення і тепер підключений до листка або v-листового вузла. Процес класифікації топології починається з методу зондування. Спочатку всі вузли до виявлення знаходяться в невиявленому стані. Спочатку контролер ПВМ буде розсилати повідомлення своїм безпосередньо підключеним підлеглим. Вузол-одержувач відповість джерелу відлунням. Він допомагає визначити час туди і назад, а також містить інформацію, яка дозволяє вузлу оголошувати своїм сусідам, до якого вузла він приєднався. Приймаючий вузол почне ретранслювати повідомлення іншим сусідам на всіх інтерфейсах, крім того, на якому він отримав запит. Залежно від стану приймального порту, будь-то резервний порт або нерезервний порт, вузли будуть діяти по-різному. У першому випадку вузол, який отримує повідомлення, буде встановлений вузол, який надіслав запит як батьківський, і в останньому випадку приймаючий вузол відкине запит. Це робиться, щоб уникнути відправки зайвої інформації вузлами, які вже приєдналися до дерева. Таким чином, мережа буде поступово створювати дерево, а вузли, які приєдналися до дерева, будуть періодично відправляти сусідню інформацію контролеру ПВМ через свого батька. Поки що цей метод слідує зондувальному методу для створення карти мережі. При виході з ладу вузла в мережі сусіди будуть вживати різних заходів відповідно до стану своїх портів.

Якщо будь-який резервний, дочірній або обрізаний порти вийдуть з ладу, про несправність буде безпосередньо повідомлено контролеру ПВМ, оскільки їх батьківський порт все ще активний і є частиною дерева управління. Якщо батьківський порт виходить з ладу, то вузол повинен автономно прийняти рішення про відновлення на рівні переадресації. Оскільки вузол був відокремлений від дерева управління, він не має маршрутів до контролера. Тому, для цього потрібно спочатку знайти нового батька. Відновлення помилок починається з розсилки повідомлень на всіх інтерфейсах (крім того, який вийшов з ладу), щоб знайти новий шлях до контролера ПВМ. Якщо порти, які отримують повідомлення мають будь-які активні батьківські порти, то вони дадуть відповідь, в іншому випадку вони передадуть повідомлення іншим своїм сусідам. У цьому методі вузли будуть вживати заходів, спілкуючись зі своїми сусідами, подібно до автономних мереж. Процес автономного відновлення після відмови схожий на мету самовідновлення, введену в системі. Але, оскільки контекст визначається в ПВМ, мережа потребує центрального органу управління, який виконує важкі завдання в мережі і все ще має контроль над вузлами всередині мережі. Крім того, протокол більше орієнтований на самовідновлення, ніж на генерацію топологічної карти мережі.

Було проведено багато досліджень з класифікатором тополоігій у бездротових мережах, особливо в спеціальних лінійних сенсорних мережах, таких як метод лінійних сенсорних мереж [20]. Ці методи в основному залежать від сили сигналу для виявлення близьких сусідів. Деякі із запропонованих рішень у спеціальних лінійних сенсорних мережах, є спеціально побудованими, призначеними для лінійних бездротових мереж. Тому, досліджуватимемо дротові мережі.

Розглянемо розподілені методи для класифікації топологій. У міру збільшення розміру мережі продуктивність централізованих алгоритмів класифікації топологій знижується. Це означає, що оскільки центральна сутність повинна обробляти більше повідомлень, то час очікування для інших вузлів збільшиться, щоб отримати свою відповідь від центральної сутності, і, отже,

знизить продуктивність загальної мережі. Для більш масштабованого підходу потрібно розглянути розподілене рішення. У розподілених системах вузли можуть мати більше свободи для прийняття власних рішень і діяти більш автономно. Крім того, розподіляючи дані та обчислювальну потужність між вузлами, мережа може відчувати кращу продуктивність. У розподілених методологіях кожен вузол або вибрана група вузлів відповідає за збір, обробку та розповсюдження сусідньої інформації всіх вузлів або вибраної групи вузлів. Наприклад, за допомогою протоколів маршрутизації стану зв'язку кожен вузол може генерувати топологічну карту мережі, але існує певна межа такого підходу. Кластеризація, як відомо, є першим етапом вирішення багатьох розподілених задач. Розбиваючи проблемний простір і дані на кілька частин, можна збалансувати робоче навантаження розподіленої мережі між декількома вузлами. Навіть для централізованих керованих мереж кластеризація може бути рішенням для підвищення їх продуктивності.

2.3 Кластеризація топологій

Відповідно до централізовано керованих методів, централізована класифікація топологій може бути важким завданням у мережі через значну кількість повідомлень, якими обмінюється повторно, і значну кількість пропускної здатності, яку вона споживає. Кластеризація - це процес групування множини вузлів разом на основі подібної ознаки під одним представником кластера, який називають головою кластера. Вузли в одному кластері можуть мати подібну поведінку, схожу функціональність, ту саму мету, досяжну в певному радіусі голови кластера тощо. Це допомагає мережі розбиватися на кластери, що не перекриваються, і розподіляти збір даних, обробку даних і розподіл цих даних між кластерами. Кластеризація дає нам такі переваги, як підвищена швидкість обробки, розширюваність мережі, простіше управління, мінімізація зберігання для комунікаційної інформації, оптимізація споживання пропускної здатності і т.д. Далі класифікує алгоритмами кластеризації за

основними критеріями: синхронний або синхронний, заснований на розташуванні або не на основі розташування, стаціонарний або мобільний. Враховуючи формування вузлів у мережі, призначення мережі та особливості вузлів, буде обрана схема кластеризації.

Початковим кроком у кожному методі кластеризації є визначення характеристик базової мережі, а потім мережа перейде до наступного етапу, який є вибором голови кластера. Вибір голови кластерів може залежати від багатьох особливостей. Одним з найбільш поширених і простих методів вибору головки кластера є використання ідентифікатора пристрою. Пристрій з найменшим або найвищим ідентифікаційним номером у діапазоні k стрибків буде вибрано як голову кластера. Краще використовувати найнижчий та найвищий ідентифікатори пристрою, щоб вибрати їх голови кластерів відповідно. Але, в залежності від мережі, для вибору головки кластера може бути використаний виділення механізму на основі більш ніж одного атрибута. Метрику на основі кількох факторів для вибору голови кластерів застосовують для всіх елементів. Після того, як голови кластерів будуть обрані, вони повідомлять своїх сусідів про зміну свого стану і чекатимуть, поки вузли приєднаються до їх кластерів. Після того, як кластери будуть сформовані, вузли перейдуть до наступної фази, яка полягає в обслуговуванні кластера. Під час фази технічного обслуговування вузли можуть залишати свій кластер і приєднуватися до інших кластерів, нові вузли можуть приєднуватися або виходити з мережі. Реконструкція кластерів є витратним завданням для мережі. На це потрібен час і чимала кількість повідомлень. Тому, ключова реконструкція повинна бути виправдана ідеально, щоб не вплинути на працездатність мережі.

Мобільність засобу залежить від місцезнаходження, отриманої з інформації, швидкості та дальності передачі. Оскільки вузли, як правило, періодично надсилають пакети маяків сусідам, щоб повідомити про свою присутність, вони повідомляють про присутність один одного, використовуючи всі перераховані вище показники, середній термін дії посилання. Буде розраховано час між двома мобільними вузлами до того, як розрахувати закінчення терміну дії посилання.

Кількість сусідів засобу важлива при виборі голови кластера, оскільки запропоноване рішення фокусується на зменшенні кількості малих кластерів. Кожен вузол засобу має заздалегідь визначений метричний поріг закінчення терміну дії. Якщо обчислений термін дії посилання перевищує попередньо визначений поріг метрики закінчення посилання, вузол оголосить про себе як про голову кластера. Перш ніж оголосити про це іншим сусідам, буде сформована виграшна метрика на основі середнього показника терміну дії посилання, розрахованого раніше, і кількості сусідніх засобів. Виграшна метрика буде транслюватися на сусідні засоби в діапазоні мовлення вузла. Приймальні вузли мають один з двох наступних станів. Вузол вже заявив про себе як про голову кластера і транслював свою виграшну метрику іншим сусідам. Вузол не заявив про себе як про голову кластера. В обох пунктах вузол буде постійно слухати вхідні виграшні показники з інших вузлів. Що стосується вузлів, які знаходяться в першому стані, якщо їх виграшна метрика менше отриманої, то вони оновлять значення свого виграшного стану до отриманого значення, а що стосується вузлів в останньому стані, то вони встановлять свою виграшну метрику на отриману. Приймальні вузли потім почнуть транслювати виграшну метрику голови кластера своїм сусідам, щоб переконатися, що всі вузли на відстані від голови кластера отримають виграшну метрику. Після того, як вузли передадуть інформацію іншим сусідам, будь-який вузол, який бажає приєднатися, відповість голові кластера пакетами збіжності, включаючи ідентифікатор пристрою, ідентифікатор голови кластера та сусідній список. Процес відправки пакетів конвергенції починається з найдальших вузлів, які називають вузлами кластер-кордону. Кожен вузол чекає, поки отримає інформацію від вузлів, які знаходяться далі від голови кластера, і після отримання їх пакетів збіжності, вузол оновить інформацію про кластер і відправить пакети збіжності на голову кластера. Отримавши пакет збіжності з усіх вузлів, головка кластера згенерує повний список сусідніх вузлів і відправить його назад до вузлів всередині свого кластера. Голова кластера забезпечить локальну таблицю підключення до вузлів. Таблиця зв'язків містить наступну інформацію: відстань до голови кластера, сусіди кожного вузла, шлюз до сусідніх кластерів. Вузли-

члени використовуватимуть маршрутизацію стану зв'язку протоколу для формування карти топології кластера на основі інформації з таблиці зв'язків. Вузли-члени також згенерують таблицю між маршрутизації, використовуючи найкоротший шлях Дейкстри, використовуючи інформацію з таблиці зв'язку з іншими кластерами. Автономні вузли не мають попередніх знань про мережу. Запропоноване рішення вбудовує в пристрій метричний поріг закінчення зв'язку та діапазон голови кластера на основі характеристик мережі та пристрою. Отже, це суперечить меті проектування автономних мереж. Алгоритм кластеризації з одним стрибком, який дозволяє вузлам вирішувати ролі самих себе без будь-якого контролера. Алгоритм є однострибковим, нелокаційним, асинхронним в схемі кластеризації, яка також може підтримувати мобільну поведінку у вузлах. Кожна голова топології вирішить, чи є вона головою кластера, виходячи з власної ваги та ваги сусідів. Найважчим вузлом серед сусідів, що знаходяться рівно в одному стрибку, буде вважатися голова грона. Кожен вузол проходить початкову фазу і визначає, чи є він головою кластера. Вузол повідомить своїм сусідам, якщо це головка кластера. Автономні вузли з більш важкими сусідами будуть чекати, щоб отримати оновлення від своїх більш важких сусідів, щоб перевірити, до якого важчого вузла приєднатися, якщо один або кілька важчих сусідів оголосили себе головами кластерів. Це рішення приймається після отримання оновлень від усіх їхніх важчих сусідів. У ситуації, коли ніхто з більш важких сусідів не оголошує себе головами кластерів, і вони хочуть приєднатися до інших кластерів, вузол оголосить про себе головою кластера. Після будь-якої зміни ролей вузлів сусідам будуть відправлятися повідомлення про оновлення. Зміни включають вузол, який оголошує себе головою кластера, вузол, що приєднується до кластера, і вузол, який знає про збій зв'язку або нові зв'язки. Після того, як кожен вузол успішно вирішує свою роль, він переходить до фази обслуговування. Кожен вузол буде слухати оновлення від ролей своїх сусідів або перевіряти їх доступність. Після отримання будь-яких оновлень на етапі технічного обслуговування може статися одна з ситуацій, згаданих нижче. Наприклад, на рис. 2.2 зображено такий випадок.

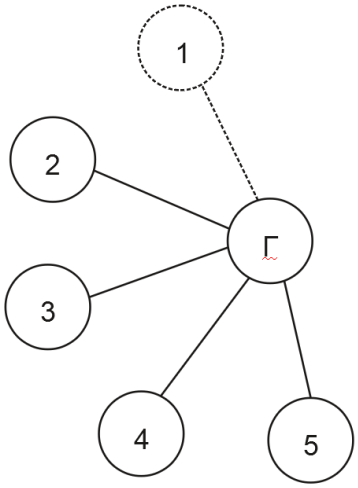


Рисунок 2.2 – Приклад зв’язку між вузлами

2.4 Організація доступу до топологічної карти мережі

Наявність доступу до топологічної карти мережі є важливою частиною управління мережею. Наявність оновленої версії карти топології дозволяє адміністратору, вузлам і будь-якому об'єкту управління в мережі приймати більш швидкі і ефективні рішення. Такі рішення, як відновлення збоїв, оновлення таблиць маршрутизації, застосування політик тощо, є наслідком дій, які залежать від наявності доступу до оновленої карти топології. Тому, більш детально значення класифікації топології в управлінні мережею потребує розгляду. Необхідно дослідити, який тип інформації повинні надавати вузлам, як її надавати і як автономні вузли будуть використовувати цю інформацію.

Автономна мережа буде самоналаштовуватися, самозахищатися, самовідновлюватися і самооптимізуватися. Ці характеристики можуть прямо залежати від топології мережі. Наприклад, аналогічно описаному методу класифікації топології, якщо посилання на певний вузол опускається вниз, автономний вузол перейде в стан відновлення після збою. Якщо відключений автономний вузол добре розуміє топологію мережі і має кілька варіантів на вибір, він може шукати можливі варіанти вузлів, які мають шлях з найменшими витратами для контролера, і оновлювати маршрутизацію відповідно до

інформації, яку він отримав з карти топології. Знання топології мережі тут дозволить знизити вартість проходження повідомлень між автономним вузлом і контролером. Згаданий приклад є одним з багатьох прикладів, які показують, як правильна класифікація топології може допомогти автономним мережам підвищити їх продуктивність.

На цьому етапі наступним кроком є пошук способу ефективної збору, обробки та розповсюдження топологічної інформації. Централізовано керовані методи, швидше за все, будуть використовуватися для цілей класифікації топології, таких як простіші у впровадженні та обслуговуванні. Однак такі методи вимагають, щоб кожен вузол, незалежно від того, скільки стрибків відходить, повинен бути на зв'язку з контролером мережі. Тому важливо запропонувати рішення, яке багато в чому залежить від спілкування з сусідами. Підхід, який пропонується, повинен намагатися мінімізувати кількість повідомлень, якими обмінюються для цілей. Зменшуючи кількість повідомлень між засобами класифікації топології, можна використовувати меншу пропускну здатність для цілей контролю та управління і залишити більшу частину пропускної здатності для передачі даних. Зменшуючи час сходження всіх вузлів до одного стану щодо карти топології, всі вузли можуть швидше приймати рішення і не стикатися з конфліктами. Іншим важливим аспектом є дослідження того, як повинна зберігатися карта топології. Більшість методологій класифікації топологій зберігають карту топології в центральному вузлі. Однак централізоване зберігання карти топології у вегетативних мережах може призвести до зниження ефективності мережі та більшого споживання пропускної здатності. Тому, необхідно знайти оптимальне рішення для способів зберігання карти топології і в автономних мережах. Всі цілі зосереджені на підвищенні ефективності мережі та залишенні більшої кількості ідентифікаторів для передачі даних.

Автономна мережа - це інфраструктура, яку використовуємо з метою генерувати карту топології мережі за допомогою засобів, наданих інфраструктурою. Відповідно до відомих методів це може бути важким завданням, бо у мережі значна кількість повідомлень, якими обмінюються та

значна кількість пропускної здатності, яку вона споживає. Підходи до класифікації топології повинні бути розроблені на основі характеристик базової мережі. Для цілей цього дослідження автономна мережа становить інфраструктуру, розробку якої потрібно задати мінімально можливим набором елементів та компонентів. Автономні елементи мережі постійно спілкуються зі своїми безпосередньо пов'язаними сусідами для обміну інформацією, такою як параметри конфігурації тощо. Тому, більш локальний підхід до збору топологічної інформації кожного вузла є морально доцільним. Однак, в повністю працюючій автономній мережі наявні складові, які відповідають за аспекти безпеки мережі. Після приєднання до домену кожен вузол повинен зв'язати реєстратора через проксі-сервер приєднання, щоб перевірити його справжність. Тому, в повністю працюючій автономній мережі вузли знають про наявність реєстратора, центрального суб'єкта, відповідального за безпеку мережі. Розробимо рішення як централізовані, так і децентралізовані рішення для класифікації топологій в автономній мережі. Мережа є стаціонарною на етапі налаштування топології, але під час фази функціонування вузлам дозволяється залишати або приєднуватися до мережі. Щоб мати можливість реалізувати рішення та спілкуватися між вузлами, використано імплементацію GRASP, написану на мові програмування Python компанією. Запропоноване рішення для класифікації топологій базується на кластерному підході. Мета тут полягає в тому, щоб згрупувати вузли в кластери, що не перекриваються, і представити вузол від кожного кластера як його представника. Визначення кластеризації та різних методів кластеризації використано для вибору пропонованого рішення. Для цілей роботи використано схему кластеризації, подібну до описаних методів кластеризації, яка є однострибичною, асинхронною, нелокаційною. Щодо аспекту мобільності схеми кластеризації, то пропонується гібридна модель. Залежно від стану вузлів, алгоритм кластеризації вважає мережу стаціонарною на етапі налаштування кластеризації та підтримує вихід або приєднання вузлів на етапі обслуговування. Описана схема кластеризації є однострибковою, оскільки автономні вузли вже знають адреси своїх сусідів і залежать від оновлень, які вони

отримують від сусідів. Оскільки автономні вузли працюють незалежно, розглянемо їх як асинхронні. Фактичне розташування вузлів (наприклад, отримане з GPS) не є фактором у процесі класифікації топологій, і фізичне з'єднання важливе. Таким чином, запропоноване рішення кластеризації для класифікації топології є нелокаційною схемою кластеризації. Після того, як кожен вузол приєднується до мережі, АПУ може надавати автономним функціям інформацію про сусідів. АПУ використовує протокол IPv6 для надання списку всіх фізично підключених сусідів і поверне список адрес всіх сусідів для кожного вузла. АПУ також може надати адресу сусідів автономному вузлу. На ранній стадії, коли вузол завантажується, вузол має попередню інформацію про своє оточення, але не знає нічого, крім своїх безпосередньо пов'язаних сусідів. Для того, щоб дозволити вузлам обмінюватися списком сусідів, зберігаємо список сусідів, наданий АПУ, як частину цінності цілі GRASP. Переговорний прапор цієї мети втілюється в дію після прийняття рішення про ААО. Поряд з сусідньої інформацією кожному вузлу присвоюється вага. Вага кожного вузла може залежати від різних атрибутів, таких як кількість інтерфейсів, агрегована пропускна здатність тощо. Кількість інтерфейсів (із зазначенням кількості прямих сусідів) вибирається в якості метрики для розрахунку ваги автономних вузлів. Наявність більших кластерів зменшить кількість головок кластерів. Тому, це відбувається в простішій кластеризованій мережі та ефективнішій міжкластерній комунікації. Вага вузла не змінюється за час його існування в мережі. Інші атрибути, пов'язані з кластером, зберігаються разом зі списком сусідів та вагою, такими як номер порту, присвоєний кожному об'єкту, зареєстрованому в ААО, список членів кластера вузла (якщо вузол обраний як кластер), локатор голови кластера вузла (якщо вузол не є кластером) та прапорець, що вказує на те, що вузол є кластером. Вся інформація, пов'язана з вузлом, зберігається у змінній, локальній для кожного вузла. Змінною є контейнер карти, який називається інформацією про вузол. Ключем карти (словника) є ім'я атрибуту вузла, а значенням цих ключів є значення атрибутів вузла. Кількість інтерфейсів з випадковим числом дозволяє мати кращі тести та експерименти. Спочатку для голови кластера

встановлено значення, але після початкової фази кластеризації значення змінюється. Пізніше, перевіривши значення, сусіди зрозуміють, що вузол є головою кластеру. Якщо вузол приєднується до кластера, значення розташування голови кластера зміниться на адресу голови кластера. Голова кластера зберігатиме члени свого кластера у списку набору кластерів. Вузол буде зберігати оновлений список своїх сусідів у списку сусідів. Списки сусідів ведуться АПУ і завжди оновлюються. Номер порту буде присвоєно кожній цілі GRASP після реєстрації в ААО. Номери портів спочатку встановлюються на 0, але в міру реєстрації цілей їх відповідні номери портів також оновлюватимуться. Після реєстрації цілі її значення може бути змінено та оновлено. Після реєстрації цілі на будь-якому ААО цій меті присвоюється випадковий номер порту. До тих пір, поки об'єктив реєструється на одному і тому ж ААО і на активному автономному вузлі, цей номер порту не змінюється. Після того, як буде зареєстровано на ААО, значення його номера порту спочатку буде зберігатися в атрибуті node info. Потім значення топології буде оновлено шляхом серіалізації інформації про вузол. На цьому етапі визначаються та створюються локальні можливості та цілі, необхідні для комунікації.

Висновки до розділу 2

Здійснено вибір топологій, обгрунтовано використання стандарту IPv6 для реалізації виконання завдання. При організації доступу до топологічної карти мереж було використано попередньо кластеризацію вузлів в мережі, що дало змогу отримати прикінцеві елементи топології «сніжинки».

3 РЕАЛІЗАЦІЯ ІНФРАСТРУКТУРИ ЗГІДНО ЗМІНЮВАНОЇ ТОПОЛОГІЇ

3.1 Графове задання розробленого рішення щодо топології інфраструкутри

Розгляд інфраструктури розпочнемо з аналізу подання скінченим автоматом, який зображено на рис. 3.1, в якому демонструються різні стани вузла під час процесів. Автомат стану, наведений нижче, показує шлях, який проходить кожен вузол.

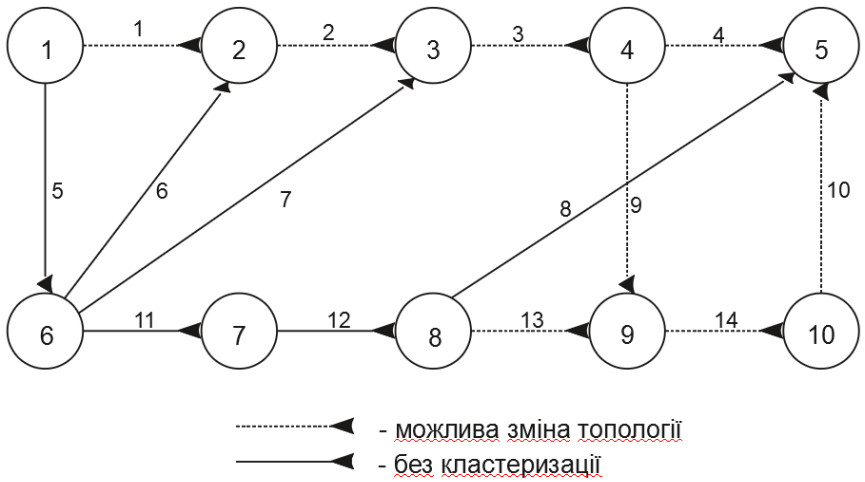


Рисунок 3.1 – Граф інфраструктури

Числа вказують на компоненти інфраструктури, а гілки в графі відносяться до подій. Після того, як вага всіх сусідів буде отримана за допомогою переговорів GRASP, вузол запустить початкову процедуру, щоб вирішити свою роль. Якщо є більш важкі сусіди, вузол переміститься в наступний стан, щоб слухати оголошення кластерів від своїх сусідів. Після того, як вага всіх сусідів буде отримана за допомогою переговорів GRASP, вузол запустить початкову процедуру, щоб вирішити свою роль. Якщо вузол є найважчим вузлом серед сусідів, вузол перейде до наступного стану, щоб оголосити про зміну своєї ролі. Всі важчі вузли самі приєднуються до іншого кластера, вузол представляється як голова кластеру. Вузол є головою і намагається виявити інші голови кластерів. Вузол приєднується до кластера, і сусіди знають, що він приєднався до якого

кластера. Вузол є кластером і переходить у фазу технічного обслуговування, щоб почати слухати будь-які типи оновлень, наприклад, збій посилань, виявлення нових посилань тощо. Вузол не є кластерною головою і переходить у фазу технічного обслуговування, починає слухати будь-які типи оновлень, наприклад, збій посилань, виявлення нових посилань тощо. Помилка зв'язку з головою кластеру знову переходьте до пошуку нового стану кластера. Вузол не є головою кластеру і повідомляє про оновлення його кластера. Вузол не є кластером і після звітності повертається до фази технічного обслуговування. Вузол є кластером і помічає оновлення. Вузол є головою кластеру і після повідомлення про оновлення кластерів і некластерних голів повернеться до фази технічного обслуговування.

Перед початком процесу кластеризації кожен вузол обмінюється своєю вагою зі своїми сусідами за допомогою переговорів GRASP. На цьому етапі налаштування кластера вузли повинні бути стаціонарними. Після того, як вузол отримає всі ваги від своїх сусідів, він перейде до фази налаштування. На етапі налаштування кожен вузол автономно вирішить, чи є він кластером. Якщо вузол має найбільшу вагу серед своїх сусідів, він оголосить про себе як про кластер, повідомивши своїх сусідів про зміну його ролі. Автономні вузли відповідають за повідомлення своїх сусідів про будь-які оновлення. Цей метод оновлення називається *моделлю*. Існує два різних підходи, коли справа доходить до оновлення. У першому випадку кожна сутність у системі надсилатиме оновлення своїм аналогам після змін, тоді як для останнього підходу суб'єкт запитуватиме оновлення від своїх аналогів. Модель витягування зазвичай виконується випадково, тоді як друга модель базується *на тригері*. Спостерігач постійно перевіряє наявність змін в системах*, заснованих на тригерах*. Якщо спостерігач помітить будь-які зміни, він буде слідувати заздалегідь визначеному протоколу, щоб вжити заходів щодо внесених змін. Оскільки зміни в мережі не є детермінованими, більш розумним є дотримання підходу на основі тригера та другої моделі. При будь-яких змінах автономні вузли несуть відповідальність за повідомлення про оновлення своїм сусідам або кластеру. Після того, як вузол змінить свою роль на голову кластеру, він почне шукати інші голови кластерів,

запустивши GRASP для інших кластерів. У реалізації, передбаченій для цього рішення, кожен вузол спочатку обмінюється ваговою та вузловою інформацією, запускаючи GRASP. Після зміни його ролі на голову кластеру, нова ціль GRASP буде зареєстрована на тому ж AAO. Ця мета буде використана головами кластерів для виявлення один одного і узгодження топології карти мережі.

Після початкової фази голови кластерів оновлять своїх сусідів. Вузли, які не є головами кластерів, почнуть слухати вхідні оновлення від своїх сусідів, щоб перевірити, який з них змінив свою роль на голову кластеру. Вузли кластерів почнуть слухати оновлення, щоб відстежувати вузли, які намагаються приєднатися до них. Причина в тому, що обрана схема кластеризації є однострибковим методом кластеризації. Найважчий вузол в околицях вузлів буде обраний в якості голови кластера. Тому кожен вузол повинен чекати, поки його більш важкі вузли оголосять про себе як про кластерні голови. Кожен вузол буде вирішувати, до кого приєднатися серед своїх більш важких сусідів, які заявили про себе як про кластер. Найважчий сусід буде обраний в якості голови кластера і приєднається до нього. Приєднавшись до кластера, вузол оголосить про це оновлення всім своїм сусідам, щоб його роль була зрозуміла всім сусідам, особливо легшим сусідам. Прийняття рішень більш легкими вузлами залежить від того, чи є їх важчі вузли. Оголосивши всім сусідам, що вузол приєднався до кластера і не є кластером, інші вузли можуть видалити його зі списку можливих варіантів об'єднання кластерів. У ситуації, коли ніхто з більш важких сусідів не оголошує себе кластерами і не вирішує сам приєднатися до інших кластерів, вузол оголосить про себе як про кластер і повідомить про це своїх сусідів. У дуже рідкісній ситуації, коли вага всіх сусідніх голів кластерів однакова, метрика, яка використовується для приєднання до кластера, є розміром його кластера Некластерний вузол приєднається до головки кластера, який має менший набір кластерів. Метою методів кластеризації є максимальне збільшення розміру кожного кластера. Таким чином, приєднання до меншого кластера дозволить розподілу розмірів кластерів бути нормальним, щоб запобігти значному збільшенню розміру одного кластера порівняно з іншими кластерами. Після того, як ролі будуть

визначені, вузли перейдуть до фази технічного обслуговування. Під час фази основної оренди, залежно від їх ролі, вузли вживатимуть різних заходів щодо вжиття заходів відповідно до оновлень, які вони отримують. Повідомлення, які отримує вузол, будуть або від їх прямого сусіда, або тільки від однорангової голови кластера. Некластерні вузли отримуватимуть повідомлення лише від своїх сусідів, які також включають їх голову кластера. З іншого боку, кластери отримують повідомлення як від своїх сусідів, так і від своїх несусідських сусідів кластерів. Виходячи з інформації, закладеної в значення вхідних запитів і ролі вузла, вузли виберуть свої наступні кроки. Вузол отримує повідомлення про оновлення на етапі обслуговування від іншого вузла. Якщо він є або кластером, або його сусідом, який не є кластером, оновлення, надіслане з нього, буде збережено в контейнері карти, який називається інформацією про сусіда. Якщо він є новим сусідом і немає запису в інформації про сусіда, новий запис буде додано до відомостей про сусіда, а потім значення вхідного запиту буде збережено в інформації про сусіда. Якщо вузол є кластером, то може статися одна з наступних ситуацій. Якщо оновлення з вузла не містить жодної інформації про зміну своєї ролі з голови кластеру на не голову кластеру, вузол оновить свою карту топології відповідно до оновлення надісланого з його кластера. Однак, якщо вхідне оновлення з вузла, яке є поточним кластером, повідомляє, що вузол змінив свою роль з голови кластеру на не голову кластеру, то вузол змінить свій стан з обслуговування на знаходження нової голови кластера. Для обох згаданих контейнерів, якщо вхідний запит надходить від нового кластера або нового сусіда, для них буде створено запис у контейнерах, а потім їх вартість буде зберігатися. Зміна ролі кластера є дорогим завданням у мережі. Кластерні голови разом створюють основу мережі. Кластерні голови вибираються на основі більш стабільних характеристик, а значить вони рідше міняються ролями. Рішення для кластеризації залежить тільки від ваги сусідів. Це може призвести до неефективної кластеризації, великої кількості повідомлень, якими обмінюються тощо. Тому, для того, щоб кластер приєднався до іншого кластера, вага нового сусіда не є єдиним фактором для зміни його ролі. Якщо голова кластера

повідомляє про наявність іншого кластера, він перевірить як вагу, так і розмір його кластерної множини. Якщо обидва мають більші значення, то в такому випадку вузол відмовиться від своєї ролі кластера і приєднається до другого вузла. Цей випадок відбувається тільки для знову доданих вузлів. Тому що на початковому етапі тільки найважчий вузол серед усіх своїх сусідів оголосить про себе як голову кластеру. Тільки в цьому випадку буде дозволено співіснування двох голів кластера. Аналогічно, для того, щоб голова кластера змінила свою роль і приєдналася до іншої голови кластера, повинні бути деякі показники більше, ніж просто порівняння ваги вузлів. Розглянемо вузол як голову кластера. Після встановлення нового зв'язку між двома вузлами, сусідами стали два кластера. Якщо перший вузол важчий за другий вузол і розмір його кластерної множини більший за розмір кластерної голови першого, то перший вузол змінить свою роль на некластерний вузол і приєднається до другого. Він оголосить про приєднання шляхом надсилання оновлень усім сусідам, які включають нову голову кластеру та його попередніх членів кластерної множини, оскільки всі вони є сусідами. У випадку, якщо вищезгадана умова не буде виконана, дві голови кластерів будуть співіснувати. Після того, як голови кластерів знайдуть один одного, вони почнуть переговори про значення цілі кластера, яка містить значення поточної версії карти кожного кластера. Після першого раунду переговорів між головами кластерів кожен кластер знову перегляне список усіх виявлених голів кластерів і перевірить, чи є поточна версія карти такою ж, як та, яку раніше отримав одноранговий кластер.

Може знадобитися більше однієї ітерації, виходячи з розташування вузлів та їх відстані, щоб усі голови кластерів мали однакову карту топології. У гіршому випадку розглянемо лінійну мережу з багатьох вузлів. Оскільки використовуємо підхід кластеризації з одним стрибком, то голови кластерів - це щонайбільше три стрибки віддалятися один від одного. Відповідно до запропонованої схеми кластеризації, вузли будуть чекати, поки більш важкі сусіди оголосять про свої ролі як голови кластерів. Вузол буде чекати, поки вузли оголосять про свої ролі. На першій ітерації оновлень, вузли оголосять себе як кластерні голови, а вузли

приєднаються до них відповідно. Хоча вузол має найменшу вагу в мережі, але оскільки всі його важчі сусіди приєдналися до інших кластерів, він оголосить про себе як голову кластеру. Максимальна відстань між головами кластерів становить два стрибки. Вузли можуть бути відсортовані наступним чином, виходячи з їх ваги. Таким чином, цей тип мережі дає нам максимально можливу відстань між будь-якими двома головами кластерів, тобто три переходи в кластерному підході. У найгіршому випадку в лінійній мережі голови кластерів знаходяться на відстані трьох стрибків. Якщо є багато кластерів, знадобиться *багато* ітерацій, щоб дані останнього вузла досягли першого вузла. У цьому сценарії вважаємо процес передачі повідомлень атомарним, що означає, що вони не перериваються. При будь-яких змінах, що відбуваються в мережі, керівник кластера сам інформується про зміни, або підлеглі вузли повідомляють про це своєму кластеру. Для оновлення інших вузлів використовуємо наявну модель. Голови кластерів підтримують останню версію карти топології для кожного вузла на основі останніх переговорів. Після отримання повідомлення про оновлення, яке впливає на карту топології, голова кластеру спочатку оновить свою власну версію, а потім звіриться з усіма сусідніми головами кластерів та підлеглими. Якщо їх версії відрізняються, то голова кластеру надішле їм повідомлення про оновлення. Якщо оновлення включає підлеглий вузол, що залишає домен, то подібно до запропонованого алгоритму голова кластеру видалить його з його сусідів, перераховує і формує новий кластер. Якщо керівник кластера виходить з мережі, то знову ж таки, дотримуючись тієї ж поведінки підлеглі змінять свій стан на початковий і або шукатимуть новий кластер, або оголосять самі себе як кластери. В рамках технічного обслуговування кожен вузол відповідає за перевірку доступності своїх сусідів (сусідів або інших кластерів). Сусідська досяжність або досяжність кластерної голови може бути виконана двома способами. Кожен вузол матиме три спроби, перш ніж він дійде висновку, що його мета недосяжна. Якщо вузол не реагує на оновлення, які вузол надсилає йому, після трьох невдалих спроб вузол розпізнає, що сусід недосяжний, і видалить його зі списку сусідів або кластерів, оновлюючи топологію та сусідню інформацію та

надсилаючи оновлення доступним вузлам. Іншим методом перевірки доступності вузлів є використання повідомлень ICMP. Протоколи можуть бути використані для перевірки досяжності сусідніх вузлів, але тільки пінг може бути використаний для перевірки наявність сусідів, які знаходяться більше, ніж на одному стрибку. Подібно до попереднього методу, після трьох невдалих спроб відповісти на повідомлення ICMP, вузол буде вважатися недосяжним, топологія буде оновлена, а оновлення буде надіслано іншим доступним вузлам. Цей процес відбувається періодично на окремому потоці, в залежності від ролі вузла. Голови кластерів будуть використовувати пінгування, щоб зв'язатися з іншими кластерами, а всі вузли будуть використовувати протокол для підтримки сусіднього списку і спостереження за змінами в сусідньому список.

3.2 Розподілення рішення для побудови кластерів

Рішення, що пропонується для реалізації є розподіленим рішенням, в якому вузли не повинні були нічого повідомляти центральному контролеру, оскільки не передбачалося центрального органу управління. Перед початком будь-якого безпечного зв'язку між будь-якими двома вузлами в повністю працюючій автономній мережі необхідно виконати протокол. Він дозволяє зобов'язанням та реєстратору досягти взаємної аутентифікації та безпечно спілкуватися один з одним або іншими перевіреними автономічними вузлами в домені. Деталі протоколу опишемо сумісно з централізованим рішенням. Це рішення збирає сусідню інформацію в реєстраторі під час процесу автентифікації. Централізовано керовані мережі мають деякі переваги перед розподіленими мережами. Ними простіше керувати, контролювати, обслуговувати. У централізовано керованих мережах роль менеджера мережі або заздалегідь визначається адміністратором, або вузли в мережі повинні пройти через вибори. Вони представляють чотири нові ролі в автономній мережі: реєстратор, застава, приєднання до проксі та протокол. Користуємося перевагами реєстратора і додаємо нове завдання управління мережею – збір, обробку і розподіл карти

топології мережі. Запропоноване рішення значно впливає на кількість повідомлень, якими обмінюються для цілей класифікації топології, і може покращити ефективність. Протокол був стандартизований. Зміна його семантики призведе до невідповідної реалізації. Розглядаємо два сценарії: один з відповідною семантикою, а інший з невідповідною семантикою. Вирішувати виконавцю чи використовувати невідповідний підхід, щоб отримати переваги. У першому сценарії, де семантика не може бути змінена, етап налаштування кластеру відбувається відразу після завершення аутентифікації. На даний момент застава отримала облікові дані та сертифікати від реєстратора, і реєстратор, і застава взаємно підтвердили один одного. Після успішної аутентифікації застава стане частиною домену і завантажить його повністю робочий АПУ. Щоб зібрати інформацію кластеру у цьому сценарії, інженер зареєструє ціль під назвою для подальшого використання для цілей переговорів з нещодавно приєднаними аутентифікованими вузлами. Також буде зареєстрований на нещодавно автентифікованих зобов'язаннях після того, як вони розпочнуть свою АПУ. На реєстраторі значення містить повну топологію мережі та номер порту, на якому можна отримати доступ. На нещодавно автентифікованому вузлі голови кластеру, значення містить список сусідів, номер порту, на якому його можна досягти, і оновлений варіант топології. Спочатку значення карти топології встановлюється нульовим, оскільки застава ще не повідомлена реєстратору. До аутентифікації застава мала список сусідніх LL-адрес для кожного інтерфейсу. Однак після автентифікації він може отримати адресу своїх сусідів, запросивши її у свого АПУ. Отримавши повний список своїх сусідів, нещодавно підтверджена обіцянка встановить переговорну сесію з реєстратором, щоб надіслати свій номер. Застава зв'язується з реєстратором через проксі-сервер приєднання. Тому, він не має локатора реєстратора і повинен виявити його, запустивши GRASP. Проксі-сервер приєднання вже має локатор реєстратора, оскільки він є автентифікованим членом мережі. Якщо нещодавно автентифікована застава запускає GRASP, щоб знайти покажчик реєстратора, проксі-сервер приєднання поверне кешований запис локатора реєстратора. Повідомлення про виявлення GRASP не повинно йти до

реєстратора, і нещодавно автентифікована застава може використовувати кешовану інформацію свого сусіда, якщо кешована інформація не закінчилася. Знайшовши локатор реєстратора, нещодавно автентифікована застава розпочне сесію переговорів GRASP з реєстратором, щоб перенести список сусідів та отримати топологію мережі від реєстратора. З боку реєстратора, після отримання запиту на переговори від новоавтентифікованої застави, карта топології буде оновлюватися відповідно до запропонованої вартості ініціатором переговорів.

Припустимо, це перший раз, коли нещодавно підтверджена застава звертається до реєстратора. У цьому випадку номер порту також буде записаний і зіставлений з адресою нещодавно автентифікованої застави. Після запису всієї інформації з оферти, реєстратор надішле оновлену карту топології ініціатору переговорів. Нещодавно підтверджена застава збиратиме відповідь реєстратора та оновлення значення карти топології. Як тільки застава автентифікується і отримує відповідь на ваучер і оновлену версію карти топології, вона переходить до етапу технічного обслуговування і починає слухати події. Кожна обіцянка після автентифікації буде діяти як проксі-сервер приєднання до інших вузлів. На етапі технічного обслуговування проксі-сервер приєднання буде повідомлено про два типи подій. По-перше, заставою є спроба приєднатися до мережі та використання вузла як проксі-сервера приєднання. У цьому випадку довірена особа приєднання знає про наявність застави, але не вживатиме жодних подальших дій щодо повідомлення про цю подію реєстратору, оскільки застава ще не автентифікована, і якщо застава буде автентична, реєстратор надішле оновлену версію топології на всі вузли. Тому, надсилання оновлення реєстратору та повідомлення реєстратора про наявність нової застави вважатиметься зайвим повідомленням. Другий тип подій - це помічання присутності або відходу *автентифікованого* сусіда. У цьому випадку, оскільки немає спостерігачів, які могли б повідомити про такі випадки безпосередньо реєстратору, автентифікований вузол, повідомлений цією подією, повідомить про це реєстратору, а реєстратор обробить оновлення та надішле оновлену інформацію топології на всі автентифіковані вузли. Спочатку реєстратор може почати завантажувати оновлення, оскільки кожен вузол отримає

однакову копію карти або розсилатиме повідомлення про оновлення окремо кожному вузлу. У першому випадку кількість зайвих повідомлень збільшиться, і вузли повинні прослуховувати повідомлення тільки від реєстратора по окремому потоку. В останньому випадку вузли можуть прослуховувати будь-які вхідні повідомлення про оновлення від аутентифікованих вузлів, а не тільки від реєстратора. Подібно до ПВМ, в цьому централізованому підході реєстратор може обробляти інформацію, що надходить, і на основі топології мережі він може формувати кластери, створювати дерева управління тощо. Наприклад, зібравши список сусідів вузла, реєстратор може створити кластери і повідомити вузол, що він є головою кластера або приєднається до іншого кластера. У цьому випадку реєстратор оновить лише голови кластерів. Виконуючи обов'язки центрального менеджера мережі, необхідно зменшити кількість підключень, які реєстратор повинен встановити, і кількість запитів, які він отримує, що призводить до зменшення навантаження на реєстратора.

Друге запропоноване рішення може бути застосоване тільки в тому випадку, якщо семантика змінена. Зміна, яку потрібно зробити, стосується рівня обмінюваних повідомлень між заставодавцем та реєстратором. Застава знає адресу своїх сусідів, перш ніж отримати автентику та приєднатися до мережі. На цьому етапі застава може підготувати список, що містить інформацію про зв'язок між його інтерфейсами і сусідами. Застава повинна зіставляти LL-адреси своїх інтерфейсів з LL-адресою підключених інтерфейсів. До того часу, коли застава хоче сформувати повідомлення про запит ваучера, воно повинно включати згаданий список сусідів в запиті ваучера. Решта процесу триває, як зазвичай, протокол, поки запит ваучера не дійде до реєстратора. Обіцянка шукатиме довірену особу в своєму районі. Знайшовши проксі-сервер приєднання, застава надішле запит на ваучер проксі-серверу, щоб його можна було передати реєстратору. Реєстратор отримає запит на ваучер і витягне з нього сертифікат, а потім спробує засвідчити справжність застави, надіславши запит до вузла. Якщо вузол автентифікує сертифікат, а реєстратор автентифікує саму заставу, він витягне список сусідів, доданий до запиту ваучера. Реєстратор має зареєстрований номер, призначений для

зв'язку через карту топології між собою та іншими вузлами. Після вилучення списку сусідів, реєстратор оновить значення карти топології та значення номера порту застави. Реєстратор змінить повідомлення про відповідь ваучера, включить оновлену карту у відповідь ваучера та надішле її назад до проксі-сервера приєднання. Потім проксі-сервер приєднання передасть відповідь від реєстратора до застави. Якщо відповідь ваучера є повідомленням про схвалення, застава спочатку витягне з нього сертифікат реєстратора, а потім витягне оновлену карту топології мережі. На даний момент застава аутентифікована і одночасно має оновлену топологію мережі. Це рішення заощадить раунд комунікації між заставодавцем та реєстратором. Якщо вузол або реєстратор відхилять заставу, реєстратор проігнорує сусідський список застави. Подібно до оригінального протоколу, реєстратор надсилає невдалу відповідь аутентифікації на заставу через проксі-сервер приєднання. Після етапу автентифікації та настроювання кластеризації відбудеться етап обслуговування.

3.3 Результати експериментів

Розглянемо експериментальні результати для перевірки протоколів кластерів. Тестовий стенд налаштовано як три різні топології. Щоб перевірити результати, остаточна карта топології, що зберігається в кожному вузлі, порівнювалася з фактичною топологією після збору результатів. Три топології представлені на рис. 3.2. Оскільки лінійна топологія на рисунку, то назвемо лінійну топологію «Топологія 1». Топологія, представлена на рисунку, виглядає як літера А, тому називаємо її «Топологія 2». Аналогічно, топологія, представлена на рисунку, виглядає повернутою літерою У, тому називаємо її "Топологія 3".

Кожна топологія має свою унікальну задачу. Вузол, який отримує два неузгоджених повідомлення про оновлення з тією ж метою з кожним інтерфейсом, є високим. Асинхронна характеристика мережі може підвищити ймовірність виникнення таких конфліктів. Тому, завдання полягає в обробці

дублювання вхідних даних. Топологія 3 є комбінацією двох топологій з більш високою зв'язністю і складністю, ніж дві інші топології.

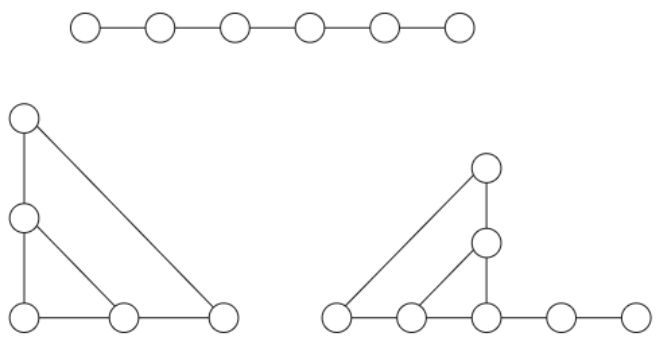


Рисунок 3.2 - Топології

Додавання внутрішнього меншого циклу в існуючу циклічну топологію може підвищити ймовірність таких небажаних подій. Усі вузли, що беруть участь у тестових стендах, працюють під управлінням операційних систем, підключені до дроту та всі включені IPv6.

Що стосується рішення кластеризації, спочатку припускаємо, що процес аутентифікації вже був виконаний, і всі вузли аутентифіковані. Тому, повідомлення щодо нього не враховуються за результатами. Схема кластеризації асинхронна. Спочатку вузли не мають попередніх знань про вагу один одного. Існує обмежена кількість методів отримання вузлом ваги свого сусіда. Вага сусіда може бути збережена в ній, витягши її з вхідного запиту на ініціювання переговорів GRASP, або вага буде вилучена з відповіді сусіда на запит про ініціювання переговорів GRASP. Ваговий обмін в даній схемі кластеризації обмежений тільки сусідами. Оскільки вони знаходяться лише на відстані одного стрибка, ця дія займає лише короткий проміжок часу, оскільки для цього не потрібно проходити маршрутизацію та ретранслюватися з одного вузла в інший.

Опишемо два методи реалізації обміну вагою. У першому способі всі вузли ініціюють запит на обмін вагою одночасно, тому що жоден з вузлів в мережі не має ніяких знань про вагу свого сусіда. Це може призвести до обміну значною кількістю повідомлень лише для обміну вагами. У другому способі деякі вузли

можуть запускатися раніше і запитувати вагу своїх сусідів раніше, ніж інші. У цьому випадку деякі вузли отримують вагу своїх сусідів, ініціюючи запит на переговори до того, як вони це зроблять, тому вони не тільки отримують вагу своїх сусідів, але й надсилають свої власні ваги як частину запиту сусідам-рівним. Це дозволяє обом колегам-учасникам мати вагу свого другого сусіда, встановлюючи єдину переговорну сесію для них. Ця коротка затримка запуску вузлів значно зменшить кількість обмінюваних повідомлень. Затримку можна змоделювати, додавши випадковий час ваги від 0 секунд до 10 секунд. Результати для запуску процесу "Одночасно" або "Затримка" відокремлюються. Результати подано в табл. 3.1.

Таблиця 3.1

Результати експерименту

Тип запуску	Топологія 1	Топологія 2	Топологія 3
Одночасно	52	58	68
Затримка	48	52	64

Кількість повідомлень, якими обмінюються в одній топології, помітно відрізняється при одночасному виконанні тестів або з невеликою затримкою. У першому сценарії, оскільки вузли не мають жодних попередніх знань про своїх сусідів, вважається, що всі вони починають вести переговори зі своїми сусідами одночасно. Однак, у другому сценарії, якщо деякі вузли мають певні знання про своїх сусідів, це допоможе їм зберегти повідомлення. Вибір вузлів, які раніше спілкувалися зі своїми сусідами, є випадковим.

Вузли або оголосять себе головами кластерів, або спробують приєднатися до кластерів. Кількість анонсів кластерів, оголошень про приєднання та обміну картами топології між головами кластерів та членами кластера в кожній топології обмежена, кількість можливих виходів також обмежена. Унікальні результати, отримані під час кожного тестового запуску, записуються в кожну клітинку таблиці. Результати групуються на основі типу топології та кількості кластерних

головок. Для деяких топологій кількість повідомлень залишається незмінною, незалежно від кількості тестових запусків. Деякі комірки залишають порожніми, оскільки в залежності від топології не має всієї кількості кластерних голів. Також, частина клітин має не один результат, який виходить в результаті багаторазових прогонів. Під час першої ітерації початкової фази кластеризації, вузли оголосять їх кластерами, оскільки вони є найважчими вузлами серед своїх сусідів і надішлють оголошення про кластерні голови. Одне оголошення кластера надсилається з першого вузла, а два оголошення кластерів надсилаються з другого. Вузол приєднається до кластера і повідомить про це своїх сусідів. Вузли об'єднуються, оскільки вони важче за них і найважче серед своїх сусідів, і відправляють три повідомлення про об'єднання в цілому. Вузол хотів приєднатися до вузла, але він не отримав оголошення про кластерну голову. Аналогічно, вузол не отримав оголошення про голову кластера від вузла. Тому, оскільки вузол не має важчого сусіда, який оголосив себе головою кластеру, він оголосить про себе як голову кластеру і надішле два повідомлення про оголошення кластера двом своїм сусідам. На заключному етапі вузол приєднається до кластеру, оскільки він є єдиним сусідом, який заявив про себе як про кластерну голову. Після того, як голови кластерів були ідентифіковані і некластерні вузли приєдналися до своїх сусідніх кластерів, кожен кластер генерує карту топології свого власного локального кластера. Потім голови кластерів намагаються виявити один одного та синхронізувати значення своєї топології, встановлюючи сесії переговорів GRASP та обговорюючи цінність карти своїх кластерів. Залежно від того, скільки кластерних голів було виявлено і як вони розподілені по мережі, потрібно один або кілька кроків для синхронізації їх значень. Максимальна кількість ітерацій, необхідна для повної синхронізації будь-якої голови кластера з іншими кластерами на етапі налаштування, залежить від того, де розташовані кластерні голови в топології. Вузли можуть мати різну кількість ітерацій для синхронізації з їх одноранговими кластерами. Під час кожної ітерації кожен кластер створить сесію переговорів GRASP зі своїми одноранговими кластерами, які були знайдені в результаті процесу виявлення GRASP.

Після того, як кластери сформовані і голови кластерів вперше синхронізували своє значення для карти топології, кожен вузол буде слухати події. Події можуть включати приєднання/вихід з мережі нового вузла або повідомлення про оновлення від підлеглого члена кластера до його ядра кластера. Встановлення нового зв'язку або видалення існуючого розглядаються як приєднання або вихід з вузла. Причина в тому, що використовуємо однопрохідну схему кластеризації. Якщо сусідній вузол недоступний за прямим посиланням, можна розглянути його вихід з мережі. Хоча він все ще може бути підключений до мережі через інше посилання і приєднатися до іншого кластеру. Щоб розглядати вузол як сусідський і живий, потрібен прямий і стійкий зв'язок між будь-якими двома сусідніми вузлами.

З метою тестування та експериментів, для об'єднання кластерних голів, додали один вузол кластера без інших вузлів у його наборі. У реальних прикладах може бути встановлений новий зв'язок між двома вузлами з окремих мереж. У цьому випадку голови кластерів з обох мереж знайдуть один одного та оновлять інформацію про свою топологічну карту щойно приєднаної мережі. Об'єднання двох мереж може призвести до додаткових ітерацій для кластерів, щоб оновити інформацію про свою топологію та поширити оновлення по всій мережі.

Запропоновані рішення мають лише одну суттєву відмінність. Якщо перше відбувається після другого, то встановлюється переговорна сесія від застави до реєстратора для обміну інформацією про кластер вузла та отримання оновленої версії топології карти. Однак, якщо перше і друге рішення відбуваються одночасно, кількість повідомлень, якими обмінюються для початкового процесу, залишається такою ж, як і для процесу. Для справедливого порівняння між двома рішеннями на основі другого рішення додано другий рядок, який містить кількість повідомлень, якими обмінювалися на етапі аутентифікації. На етапі технічного обслуговування обох рішень на базі другого рішення буде здійснюватися обмін фіксованою кількістю повідомлень, оскільки аутентифікація вже виконана. Кількість повідомлень на етапі технічного обслуговування для обох рішень така ж, як результати, наведені в першому рядку таблиці. Кожен

вузол зв'яжеться з реєстратором, щоб повідомити сусідню інформацію та оновлення. Реєстратор, отримавши сусідню інформацію або оновлення, погано відображає ці оновлення на карті топології і пересилає екземпляр на всі вузли. Залежно від того, який вузол обраний реєстратором, кількість обмінюваних повідомлень відрізняється. Повідомлення містять повідомлення про реєстрацію реєстратора та оновлення топології. Після того, як кожен вузол приєднується до мережі, після прийняття, реєстратор повинен оновити свою локальну версію карти топології та надіслати її всім аутентифікованим вузлам.

Для методу кластеризації слід зазначити, що після формування кластери повинні пройти деякі ітерації та сесії переговорів GRASP, синхронізувати їх значення з топологією карти. Повідомлення для підходу кластеризації калькулюються шляхом додавання повідомлень, якими обмінювалися під час обміну протягом тривалого часу, оголошення ролей (надсилання оголошень про голови кластерів або приєднання до оголошень), а також повідомлень, якими обмінювалися керівники кластерів та члени кластера синхронізувати їх значення з топологією карти. Виконання кожного підходу індивідуально. Першим спостереженням буде кореляція між розміром і зв'язністю топології з числом обмінюваних повідомлень, незалежно від топології. Швидкість, з якою зростає кількість повідомлень, відрізняється. Залежно від типу, призначення мережі тощо можуть використовуватися різні підходи. Тому, порівняння отриманих результатів від підходу не є переконливим доказом переваги одного підходу над іншим. Виявлення топології відіграє важливу роль в обслуговуванні мережі та управлінні мережею. Це дозволяє вузлам краще розуміти своє оточення і заздалегідь приймати такі рішення, як відновлення після збоїв. Очікується, що автономні мережі досягнуть самокерованих мереж, іншими словами, вони зможуть самовідновлюватись, самозахищатись, самоналаштовуватись та самооптимізуватись. Надання топологічної інформації мережі автономним вузлам може допомогти їм приймати рішення з меншою кількістю повідомлень. Існуючі методи поділяються на дві основні групи: централізовані підходи та децентралізовані підходи. Підхід, який буде використовуватися в тій чи іншій

ситуації, залежить від типу мережі і від того, як часто вузли в мережі вимагають інформації топології. Було запропоновано розгляд як централізованого, так і розподіленого підходів до виявлення і обслуговування топології в автономній мережі. Централізований підхід використовує переваги безпечного протоколу завантаження в автономних мережевих роботах. Він ввів в мережу ролі для виконання завдань, пов'язаних з безпекою, таких як автентифікація пристроїв. Перший метод має місце після того, коли вузли надійно досягли взаємної аутентифікації. Після досягнення взаємної аутентифікації заставодавець, тепер аутентифікований член мережі, надсилає реєстратору сусідню інформацію. Тепер реєстратор відповідає за збір сусідньої інформації зі всіх вузлів і генерування і розподіл топології карти автентичних вузлів. Інший метод вимагає модифікації шляхом додавання додаткової інформації до повідомлення. Застава приєднується до мережі, надсилаючи реєстратору запит ваучера. Якщо можна вбудувати сусідню інформацію вузла в той самий запит ваучера, то можна аутентифікувати вузол і збирати його сусідню інформацію одночасно у реєстратора. Тоді реєстратор зможе генерувати/оновлювати карту топології аутентифікованої мережі та розподіляти оновлену версію між вузлами. Друге запропоноване рішення використовує схему кластеризації. Оцінюючи інфраструктуру автономних мереж, вибрано розподілену схему кластеризації, яка дозволяє вузлам діяти автономно. Запропонована схема є однострибковою, нелокаційною та асинхронною схемою. Схема кластеризації вважається стаціонарною на етапі налаштування кластера, але вона підтримує вихід або приєднання вузлів на етапі технічного обслуговування. Кожному вузлу буде присвоєно вагу на основі показника, наприклад, кількості активних інтерфейсів у мережі. Вага буде ділитися з сусідами. Взагалі кажучи, вузли з більш важкою вагою будуть заявляти про себе як про кластер своїм сусідам. Кожен вузол, який не заявив про себе як кластер, приєднається до одного з сусідів, який представився кластером. Якщо некластерний вузол має важчих сусідів, але жоден з них не представив себе як голова кластера, то вузол оголосить про себе як про голову кластеру. Після формування кластерів голови кластерів знайдуть один

одного і спробують синхронізувати карту топології свого кластера, пройшовши кілька ітерацій узгодження. Для кластерів були вузли, такі як приєднання нових вузлів або вихід вузла з мережі. Вузол надішле оновлення до свого кластера, а голова кластера розподілить його між іншими головами кластерів та членами свого набору кластерів. Переглянувши переваги такого підходу в автономних мережах, було поставлено за мету максимально зменшити кількість повідомлень, що обмінюються для цілей, оскільки розмір мережі зростав. Запропоновані рішення забезпечують ефективний підхід до збору, генерації, оновлення та розповсюдження топологічної карти. Всі представлені рішення сумісні з визначенням автономної поведінки, наданим в RFC 7575. Було зосереджено на мінімізації кількості повідомлень, але рішення все ще можна оптимізувати, щоб ще більше зменшити кількість повідомлень на кожному етапі. На етапі утримання топології можна дозволити двом сусіднім вузлам стати кластерними головами. Кожен вузол періодично проходив фазу налаштування, щоб зменшити можливість мати занадто багато таких сусідніх голів кластерів. Тому, було зосереджено на дротових мережах і обмеженій формі мобільності, при якій вузли можуть приєднуватися і залишати мережу на етапі обслуговування топології. Крім того, всі реалізації відбувалися в дротових мережах. Сервісна топологічна карта надає вузлам інформацію про те, які автономні вузли в мережі підтримують яку мету. Зіставляючи автономні вузли з цілями, які вони підтримують, можна дозволити вузлам пропускати процес виявлення, що збереже значну кількість повідомлень від приховування пропускної здатності. Реєстратор може діяти як центральна сутність і створювати/підтримувати кластери для підвищення продуктивності мережі. Використовуючи гібридний метод, робоче навантаження мережі буде розподілено на вузли. Наприклад, налаштування кластеризації може відбуватися за допомогою обох методів, які можна скомбінувати через залежності кроків.

Висновки до розділу 3

Таким чином, запропоновано графове задання розробленого рішення щодо топології інфраструктури мережі. Для побудови мулькомп’ютерної системи було використано розподілене рішення для побудови кластерів. Для цього рішення в основу закладено централізований та децентралізовані підходи. В результаті розробено гібридне рішення для створення мультикомп’ютерної системи.

ВИСНОВКИ

Обчислення, які здійснюються з використанням сучасних комп'ютерів та багатомашинних комплексів зростають. В зв'язку з цим зростає потреба у збільшенні обчислювалних ресурсів. Крім того, недостатність стандарту IPv4 спонукатиме перехід до IPv6. Наявність багатьох вузлів в комп'ютерних мережах потребує узгодження їх взаємодії через відповідні протоколи та засоби взаємодії. Все це може бути вирішене створенням мультикомп'ютерних систем для розв'язання певного класу задач. При їх створенні було застосовано топологію «сніжинка», що найбільш адекватно відповідає наявній топології вузлів в мережах. Навколо частини прикінцевих вузлів сформовано кластери, в яких один з вузлів є головою списку вузлів в кластері. Обробка повідомлень в таких кластерах є важливою в контексті фракталів сніжинок. В результаті, було запропоновано використовувати гібридний підхід до створення мультикомп'ютерні системи на базі топології «сніжинка», який враховує як централізацію так і децентралізацію.

Виявлення топології відіграє важливу роль в обслуговуванні мережі та управлінні мережею. Це дозволяє вузлам краще розуміти своє оточення і заздалегідь приймати такі рішення, як відновлення після збоїв. Очікується, що автономні мережі досягнуть самокерованих мереж, іншими словами, вони зможуть самовідновлюватись, самозахищатись, самоналаштовуватись та самооптимізуватись. Надання топологічної інформації мережі автономним вузлам може допомогти їм приймати рішення з меншою кількістю повідомлень.

В результаті, було встановлено, що існуючі методи поділяються на дві основні групи: централізовані підходи та децентралізовані підходи. Підхід, який було використано в тій чи іншій ситуації, залежить від типу мережі і від того, як часто вузли в мережі вимагають інформації топології. Було запропоновано централізований і розподілений підходи до виявлення і обслуговування топології в автономній мережі.

Оцінюючи інфраструктуру автономних мереж, вибрано розподілену схему кластеризації, яка дозволяє вузлам діяти автономно. Запропонована схема є однострибковою, нелокаційною та асинхронною схемою. Схема кластеризації вважається стаціонарною на етапі налаштування кластера, але вона підтримує вихід або приєднання вузлів на етапі технічного обслуговування. Кожному вузлу буде присвоєно вагу на основі показника, наприклад, кількості активних інтерфейсів у мережі. Вага буде ділитися з сусідами. Взагалі кажучи, вузли з більш важкою вагою будуть заявляти про себе як про кластер своїм сусідам. Кожен вузол, який не заявив про себе як кластер, приєднається до одного з сусідів, який представився кластером. Якщо некластерний вузол має важчих сусідів, але жоден з них не представив себе як голову кластера, то вузол оголосить про себе як голову кластеру. Після формування кластерів голови кластерів знайдуть один одного і спробують синхронізувати карту топології свого кластера, пройшовши кілька ітерацій узгодження.

Запропоновані рішення забезпечують ефективний підхід до збору, генерації, оновлення та розповсюдження топологічної карти. Всі представлені рішення сумісні з визначенням автономної поведінки, наданим в RFC 7575.

У першому розділі розглянуто концепція та принципи функціонування мульикомп’ютерних систем, наведено відмінності між різними варіантами топологій, розглянуто класифікацію як метод, що може бути використано для класифікації прикінцевих вузлів в топології «сніжинка», проведено огляд відомих рішень. В результаті аналізу попередніх досліджень було встановлено необхідні методи, підходи та засоби для створення мульикомп’ютерних систем.

У другому розділі здійснено вибір топологій, обгрунтовано використання стандарту IPv6 для реалізації виконання завдання. При організації доступу до топологічної карти мереж було використано попередньо кластеризацію вузлів в мережі, що дало змогу отримати прикінцеві елементи топології «сніжинки».

У третьому розділі запропоновано графове задання розробленого рішення щодо топології інфраструктури мережі. Для побудови мулькомп’ютерної системи було використано розподілене рішення для побудови кластерів. Для

цього рішення в основу закладено централізований та децентралізовані підходи. В результаті розроблено гібридне рішення для створення мультикомп’ютерної системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Лазарович І. М. Комп'ютерні системи: конспект лекцій. Івано-Франківськ : Прикарпатський національний університет імені Василя Стефаника, 2014. 190 с.

2. Проєктування комп'ютеризованих систем управління: Опорний конспект лекцій. – Тернопіль, ТНЕУ. Доступ до ресурсу: http://dspace.tneu.edu.ua/retrieve/52377/Лекції_ПКСУ.pdf.

3. Тарарака В.Д. Архітектура комп'ютерних систем: навч. посіб. Житомир: ЖДТУ, 2018. 383 с.

4. Balta-Ozkan N., Davidson R., Bicket M., Whitmarsh L. Social barriers to the adoption of smart homes. Energy Policy. 2013. Vol 63. P. 363-374

5. Mourad R., Sinoquet C., Zhang N. L., Liu T., Leray P. A survey on latent tree models and applications. Journal of Artificial Intelligence Research. 2013. Vol. 47. P. 157–203.

6. Jernite Y., Halpern Y., Sontag D. Discovering hidden variables in noisy-or networks using quartet tests. Advances in Neural Information Processing Systems (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.). 2013. Vol. 26. P. 2355–2363.

7. Anandkumar P., Valluvan R. Learning loopy graphical models with latent variables: Efficient methods and guarantees. The Annals of Statistics. 2013. Vol. 41. No. 2. P. 401–435.

8. Asbeh N., Lerner B. Learning latent variable models by pairwise cluster comparison: Part i - theory and overview. Journal of Machine Learning Research. 2016. Vol. 17. No. 223. P. 1– 52.

9. Asbeh N., Lerner B. Learning latent variable models by pairwise cluster comparison: Part ii - algorithm and evaluation. Journal of Machine Learning Research. 2016. Vol. 17. No. 230. P. 1–45.

10. Xiao J., Liao L., Hu, J., Chen Y., Hu R. Exploiting global redundancy in big surveillance video data for efficient coding. Cluster Comput. 2015. Vol 18. P 531–540.

11. Yang M., Hua G., Feng Y., Gong J. Fault-Tolerance Techniques for Spacecraft Control Computers. Wiley. 2017. 352 p.

12. Zegzhda P.D., Alekseev I.V. Specification-Based Classification of Network Protocol Vulnerabilities. Automatic Control and Computer Sciences. 2021. Vol 54. P. 922–929.

13. Li G., Vandamme A.-M., Ramon J. Learning ancestral polytrees: Learning Tractable Probabilistic Models. The Workshop of Learning Tractable Probabilistic Models at The 31st International Conference on Machine Learning. 2014. Beijing, China.

14. Etesami J., Kiyavash N., Coleman T. Learning minimal latent directed information polytrees. Neural Computation. 2016. Vol. 28. No. 9. P. 1723–1768.

15. Zhilenkov A.A., Chernyi S.G. Enhanced Fault Tolerance in Software and Hardware Network Control Systems Using Soft Cloud Storage. Automatic Documentation and Mathematical Linguistics. 2020. Vol. 54. P. 36 - 42.

16. Albuquerque A., Caixinha D. Mastering Elixir: Build and scale concurrent, distributed, and fault-tolerant applications. 1st edition; Publishing, 2018; p 574.

17. Ujile A., Ding Z. A dynamic approach to identification of multiple harmonic sources in power distribution systems. International Journal of Electrical Power & Energy Systems. 2016. Vol. 81. P. 175–183.

18. Avoine G. Carpent X. Kordy B. Tardif F. How to Handle Rainbow Tables with External Memory. ACISP 2017: Information Security and Privacy. Lecture Notes in Computer Science, May 31, 2017; Pieprzyk, J., Suriadi, S. Eds.; Springer, Cham. 2017. Vol. 10342. P. 306–323.

19. Bao Z., Dinur I., Guo J., Leurent G., Wang L. Generic Attacks on Hash Combiners. Journal of Cryptology. 2020. Vol. 33. P. 742–823.

20. Blanke M., Kinnaert M., Lunze J., Staroswiecki M. Diagnosis and Fault-Tolerant Control., 3rd ed., Springer-Verlag: Berlin Heidelberg. 2016. 695 p.

1. 21. Volkanov D.Y. Method for Choosing a Balanced Set of Fault-Tolerance Techniques for Distributed Computer Systems. Automatic Control and Computer Sciences. 2018. Vol. 51. P. 539–550.

22. Du D., Xu S., Cocquempot V. Observer-Based Fault Diagnosis and Fault-Tolerant Control for Switched Systems. Series: Studies in Systems, Decision and Control. Springer: Singapore. 2021. Vol. 280. 81 p.

23. Dankers A. F., Van den Hof P. M., Bombois X., and Heuberger P. S. Identification of dynamic models in complex networks with prediction error methods: Predictor input selection. IEEE Transactions on Automatic Control. 2016. Vol. 61. P. 937–952.

24. Shen Z., Wang W.-X., Fan Y., Di Z., Lai Y.-C. Reconstructing propagation networks with natural diversity and identifying hidden sources. Nature communications, 2014. Vol. 5. P. 4323.

25. Brunton S. L., Proctor J. L., Kutz J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences. 2016. Vol. 113. No. 15. P. 3932–3937.

26. Nitzan M., Casadiego J., Timme M. Revealing physical interaction networks from statistics of collective dynamics. Science Advances. 2017. Vol. 3. No. 2.

27. Dimovska M., D. Materassi. Granger-causality meets causal inference in graphical models: Learning networks via non-invasive observations. IEEE 56th Annual Conference on Decision and Control (CDC). 2017. P. 5268–5273.

28. Eichler M. Granger causality and path diagrams for multivariate time series. Journal of Econometrics. 2017. Vol. 137. No. 2, P. 334 – 353.

29. . Su R.-Q, Wang W.-X., Lai Y.-C. Detecting hidden nodes in complex networks from time series. Physical review E. 2022. Vol. 85. No. 6. P. 065201.

30. Deka D., Baldick R., Vishwanath S. One breaker is enough: hidden topology attacks on power grids. Power & Energy Society General Meeting. 2015 IEEE. P. 1–5.

31. Nozari E., Zhao Y., Corte´s J. Network identification with latent nodes via autoregressive models. IEEE Transactions on Control of Network Systems. 2018. Vol. 5. No. 2. P. 722–736.

32. Kivits E., Van den Hof P. M. On representations of linear dynamic networks. IFAC-PapersOnLine. 2018. Vol. 51. P. 838 – 843.

33. Yuan Y., Stan G.-B., Warnick S., Goncalves J. Robust dynamical network structure reconstruction. Automatica. 2021. Vol. 47. No. 6. P. 1230–1235.

34. Yue Z., Thunberg J., Pan W., Ljung L., Goncalves J. Linear dynamic network reconstruction from heterogeneous datasets. IFAC-PapersOnLine. 2017. Vol. 50. No. 1. P. 10586– 10591.

35. Choi M. J., Tan V. Y., Anandkumar A., Willsky A. S. Learning latent tree graphical models. Journal of Machine Learning Research. 2021. Vol. 12, P. 1771–1812.

36. Zorzi M., Sepulchre R. Ar identification of latent-variable graphical models. IEEE Transactions on Automatic Control. 2016. Vol. 61. P. 2327–2340.

37. Barrett S.F., Pack D. J., Thornton M. A. Microchip AVR®Microcontroller Primer: Programming and Interfacing. Morgan & Claypool Publishers. 2019. 374 p.

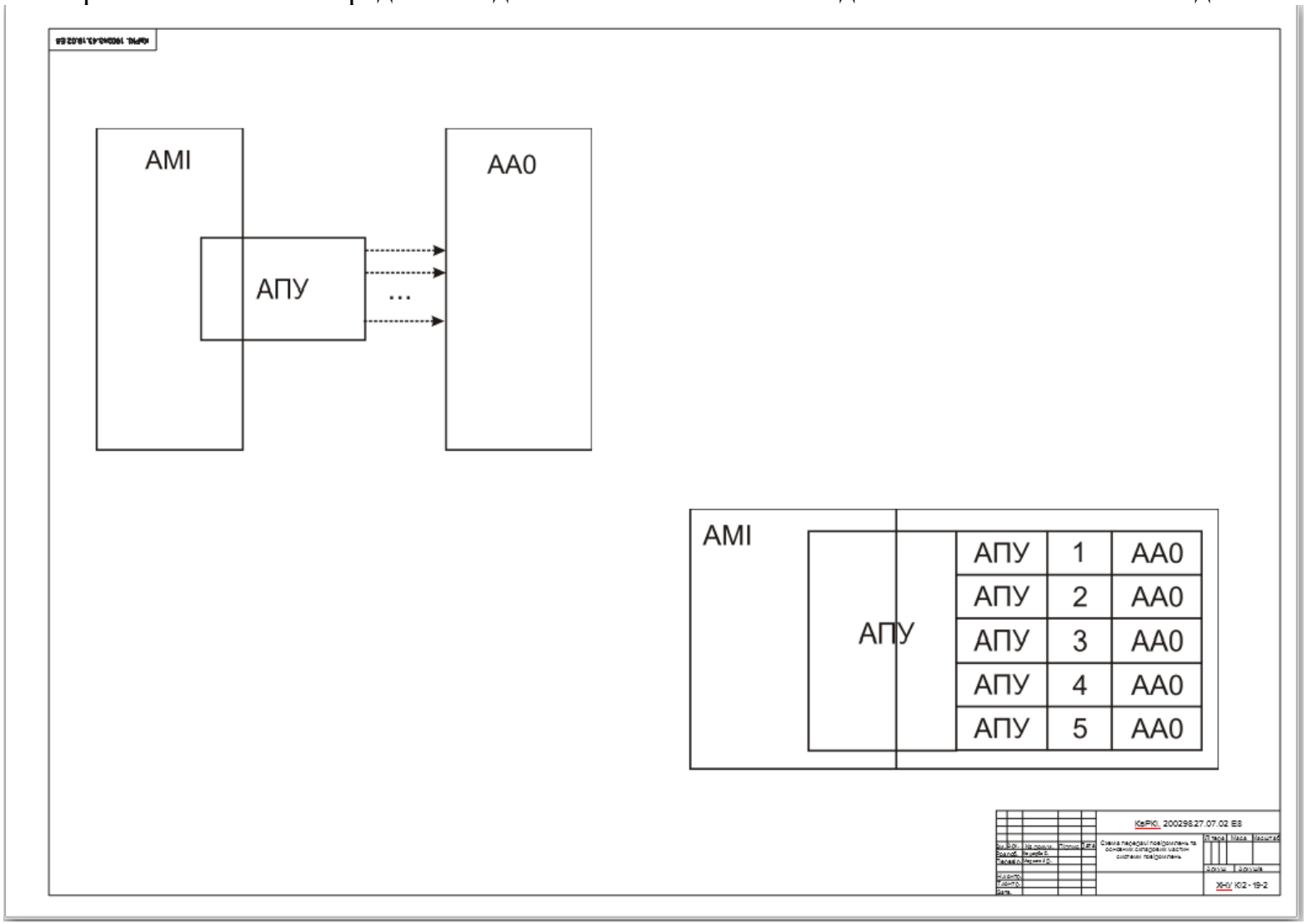
38. Kravets A.G., Bolshakov A.A., Shcherbakov M.V. Cyber-Physical Systems: Industry 4.0 Challenges (Studies in Systems, Decision and Control, 260). 2020. 349 p.

39. Yadin A. Computer Systems Architecture. *Chapman and Hall/CRC*. 2016. 467 p.

40. Anandkumar K., Chaudhuri D. J., Hsu S. M., Kakade L., Song L., Zhang T. Spectral methods for learning multivariate latent tree structure. *Advances in Neural Information Processing Systems*. 2021. P. 2025–2033.

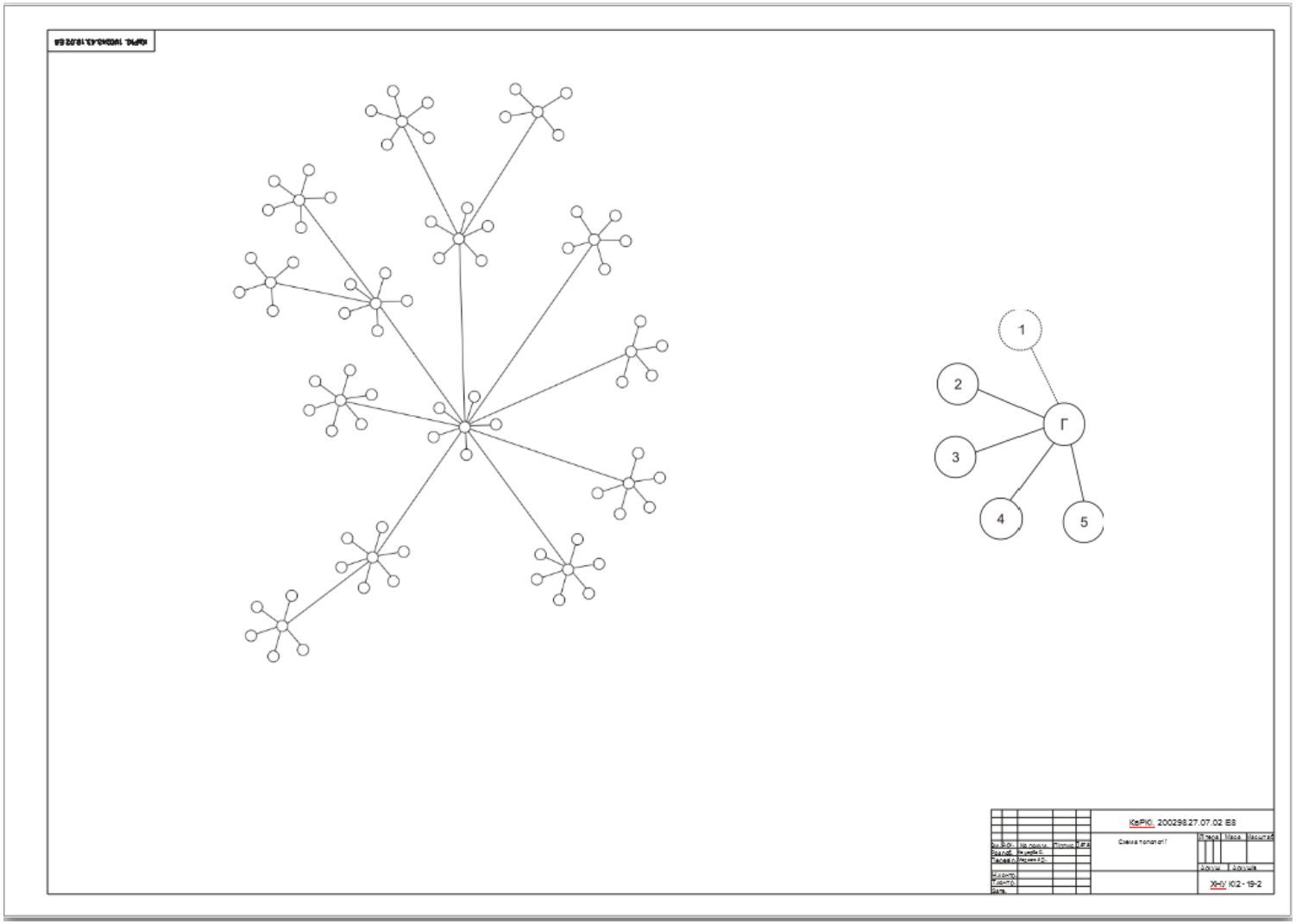
ДОДАТОК А

Копія креслення «Схема передачі повідомлень та основних складових частин системи повідомлень»



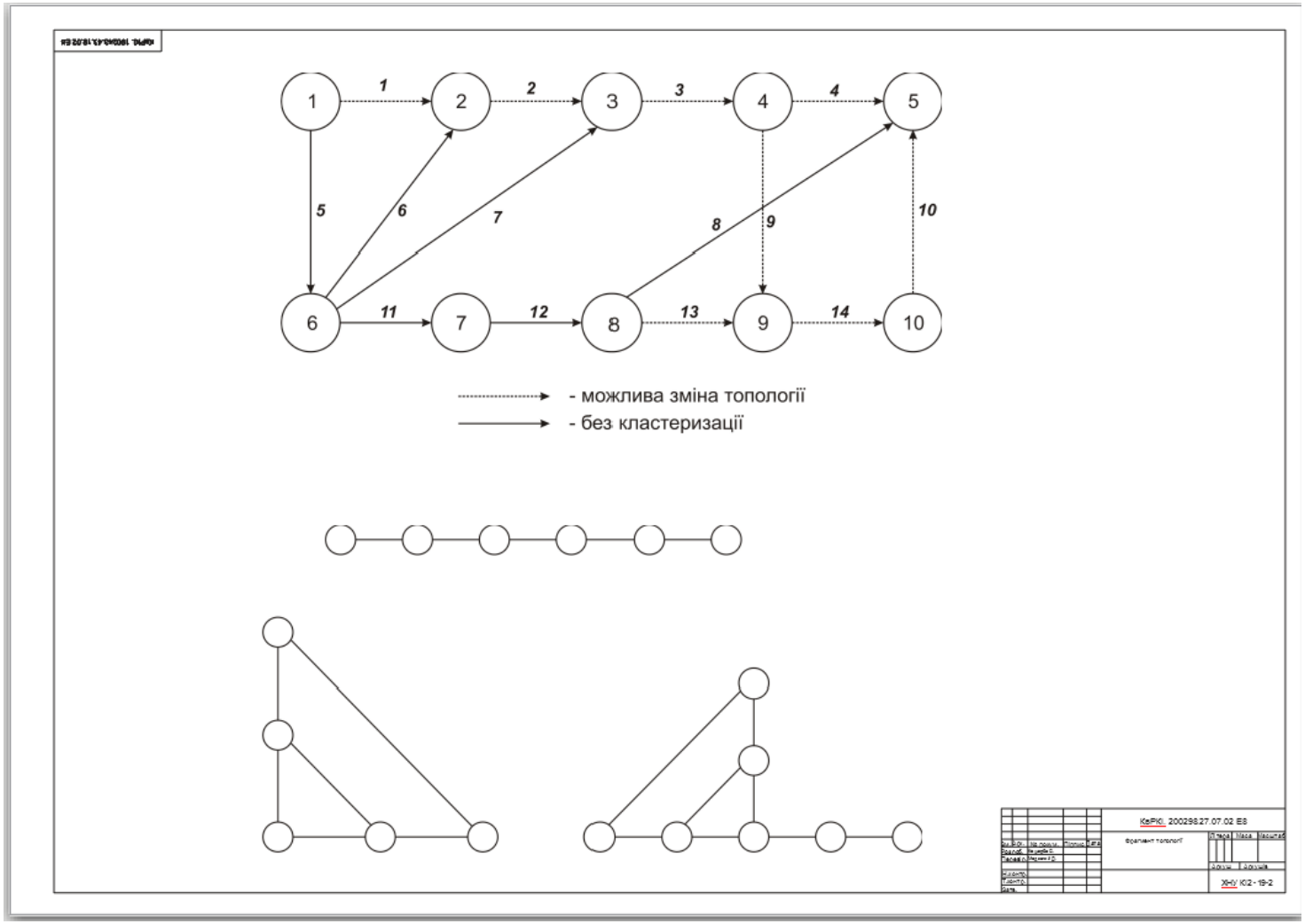
ДОДАТОК Б

Копія креслення «Схема топології»



ДОДАТОК В

Копія креслення «Фрагмент топології»



Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114272 Назва: БКР Мультикомп'ютерна система згідно топології «сніжинка» Додано в БД: 2023-05-30 Автора: С. В. Коцюрба Керівники: Д. М. Медзатий Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	113228	949	1063 (1%)	14 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача: Кафедра КІ

ID перевірки: 1015312735

Дата перевірки: 30.05.2023 10:28:45 EEST

Тип перевірки: Doc vs Internet + Library

Дата звіту: 30.05.2023 10:34:16 EEST

ID користувача: 100005591

Назва документа: Коцюрба_Мультикомп'ютерна система згідно топології «сніжинка»

Кількість сторінок: 76 **Кількість слів:** 15397 **Кількість символів:** 115123 **Розмір файлу:** 463.00 KB **ID файлу:** 1014983753

4.34%
Схожість

Найбільша схожість: 1.77% з джерелом з Бібліотеки (ID файлу: 1014517653)

3.3% Джерела з Інтернету 297 ..... Сторінка 78

1.93% Джерела з Бібліотеки 100 ..... Сторінка 80

0.23% Цитат

Цитати 1 ..... Сторінка 81

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 2

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Коцюрби Сергія

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-19-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

31 травня 2023 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Синтез та моделювання операційного автомату на основі автомату Мура

Автор: Кацюрба Сергій

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Медзатий Дмитро Миколайович, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-30 джерелами на один фрагмент речення;
2) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.(Тут текст можна і треба модифікувати)

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 4,34% і адресується до 497 першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру технічного завдання і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Дмитро МЕДЗАТИЙ

Гарант ОП _____ Сергій ЛИСЕНКО

Завідувач кафедри КІС _____ Тетяна ГОВОРУЩЕНКО

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Сергій КОЦЮРБА

Тема: Мультикомп'ютерна система згідно топології «сніжинка»

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 56

1. Короткий зміст роботи та прийнятих рішень У роботі розроблено мультикомп'ютерну систему згідно топології «сніжинка»

2. Висновок про відповідність роботи дипломному завданню Дипломна робота відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі розглянуто концепція та принципи функціонування мультикомп'ютерних систем, наведено відмінності між різними варіантами топологій, розглянуто класифікацію як метод, що може бути використано для класифікації прикінцевих вузлів в топології «сніжинка». У другому розділі здійснено вибір топологій, обгрунтовано використання стандарту IPv6 для реалізації виконання завдання. У третьому розділі запропоновано графове задання розробленого рішення щодо топології інфраструктури мережі. У Висновках підведено підсумки виконаної роботи.

4. Позитивні сторони роботи: Розроблена мультикомп'ютерна система згідно топології «сніжинка».

5. Негативні сторони роботи: немає.

6. Оцінка графічного оформлення та пояснювальної записки роботи: __-____

7. Відгук про роботу в цілому: __Робота виконана на належному рівні.

8. Інші зауваження: ___-

9. Оцінка дипломної роботи:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи вважаю, що робота заслуговує оцінки «добре» 4,00 (С)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) __Мартинюк Валерій Володимирович, д.т.н., професор, завідувач кафедри АКІТР ХНУ

"__31__" __травня__ 2023р.