Державний торговельно-економічний університет

Кафедра комп’ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка імітаційної моделі передачі даних в мережі»

Студента 4 курсу, 8 групи, спеціальності 122 «Комп’ютерні науки» Гончарук Денис Ігорович

_____

підпис студента

Науковий керівник старший викладач кафедри Селіванова Анна Віталіївна

_____

підпис керівника

Гарант освітньої програми кандидат технічних наук, доцент Демідов Павло Георгійович

_____

підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022 р.

Завдання
на випускну кваліфікаційну роботу студенту
Гончарук Денис Ігорович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи
«Розробка імітаційної моделі передачі даних в мережі»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: Розробка імітаційної моделі передачі даних в мережі
Об'єкт дослідження: процеси передачі даних в мережі
Предмет дослідження: інформаційні технології моделювання передачі даних в мережі
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Селіванова А.В.		

2	Селіванова А.В.		
3	Селіванова А.В.		

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Загальна проблематика передачі даних в мережах

1.1. Теоретичні основи передачі даних в мережі

1.2. Технології передачі даних

1.3. Моделі передачі даних

РОЗДІЛ 2. Розробка моделі передачі даних в мережі

2.1. Проектування схеми мережі передачі даних

2.2. Специфіка програмної реалізації

2.3. Програмні середовища моделювання передачі даних в мережі

РОЗДІЛ 3. Реалізація імітаційної моделі передачі даних в мережі

3.1. Специфіка побудови імітаційних моделей передачі даних в мережі

3.2. Реалізація імітаційної моделі передачі даних в середовищі моделювання Cisco Packet Tracer. (середовище можете обрати на власний розсуд)

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Дата видачі завдання

8. Керівник випускної кваліфікаційної роботи

Селіванова А.В.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент-дипломник

Гончарук Д.І.
(прізвище, ініціали, підпис)

11. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Керівник випускної кваліфікаційної роботи

____
(підпис, дата)

12. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Гончарук Д.І.
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми____________________________ Демідов П.Г. ____
(підпис, прізвище, ініціали)

Завідувач кафедри ____________________________________ Пурський О.І. ____
(підпис, прізвище, ініціали)

«_____»_________________2023 р.

ЗМІСТ

ВСТУП..... 6
1.ЗАГАЛЬНА ПРОБЛЕМАТИКА ПЕРЕДАЧІ ДАНИХ В МЕРЕЖІ.....8
1.1. Теоретичні основи передачі даних в мережі 8
1.2. Технології передачі даних.....12
1.3. Моделі передачі даних 17
2.. РОЗРОБКА МОДЕЛІ ПЕРЕДАЧІ ДАНИХ В МЕРЕЖІ.....29
2.1. Проектування схеми мережі передачі даних.....29
2.2. Специфіка програмної реалізації.....33
2.3. Програмні середовища моделювання передачі даних в мережі.....34
3.РЕАЛІЗАЦІЯ ІМІТАЦІЙНОЇ МОДЕЛІ ПЕРЕДАЧІДАНИХ В МЕРЕЖІ...38
3.1. Специфіка побудови імітаційних моделей передачі даних в мережі 38
3.2. Реалізація імітаційної моделі передачі даних в середовищі моделювання Packet Tracer. 46
ВИСНОВОК 56
ЛІТЕРАТУРА 57

ВСТУП

Актуальність теми

У майбутньому ми прагнемо проаналізувати специфічний характер передачі даних, що відбувається в будь-який момент часу. Розглядаючи тонкощі процесів передачі даних у комп’ютерній мережі, ми в майбутньому називатимемо ці процеси комп’ютерною мережею.

Без комп’ютерних мереж неможливо уявити повсякденне життя чи роботу, яку виконують компанії та уряди. З кожним роком комп'ютери, які використовуються в цих мережах, стають все більш досконалими і складнішими. Вони є невід'ємною частиною галузі інформаційних технологій. Актуальним завданням є створення програми моделювання комп’ютерних мереж. Це тому, що програмістам потрібен спосіб створення робочих моделей комп’ютерних мереж для використання мережевими адміністраторами та студентами інформатики. Ці програми дозволяють студентам закріплювати теоретичні знання, а адміністраторам – діагностувати проблеми.

Мета роботи.

Метою цього проекту є розробка програми, яка моделює комп’ютерні мережі шляхом моделювання.

Для досягнення мети необхідно вирішити наступне:

Щоб проаналізувати предмет, який конкретно стосується їхньої сфери знань.

Створіть мережеву комп’ютерну модель.

Створити програму, яка проектує модель для моделювання комп’ютерної мережі.

Реалізувати додаток.

Провести тестування програми.

Огляд літератури — це тип есе, в якому детально викладаються результати академічного дослідження.

У 21-й і 22-й роботах представлені основні уявлення про комп'ютерні мережі, а також виведені з них ідеальні архітектурні рішення.

Кілька джерел, наприклад №7, 8, 15–18, надають інформацію про популярні протоколи та алгоритми комп’ютерних мереж.

У цьому документі докладно описано методи створення та тестування моделей комп’ютерних мереж, достатньо великих, щоб бути практичними. Також описано підходи до створення та оптимізації моделей роботи цих мереж.

У роботах № 1, 10, 11, 19 розглядаються різні методи створення програмних емуляцій і моделей комп'ютерних мереж.

Аспекти роботи, включаючи її обсяг і структуру.

Твір містить пролог, п'ять розділів і додаток. Він містить 53 сторінки та 23 джерела літератури.

Про що йдеться у творі, як правило, жирним шрифтом.

У першій частині книги розглядаються різні підходи до моделювання комп’ютерних мереж і розвінчуються деякі помилкові теорії. Він також порівнює існуючі рішення з комп’ютерними мережами.

У другому розділі дисертації під назвою «Розробка модель комп’ютерної мережі» пояснюється процес проектування та розробки системи через її модель.

Третій розділ «Реалізація» детально описує архітектуру системи, спосіб її реалізації та інтерфейс користувача, а також тестування мережі.

У висновку розкриваються основні відкриття та ідеї дисертації. Він також містить вказівки для додаткових досліджень.

РОЗДІЛ 1. ЗАГАЛЬНА ПРОБЛЕМАТИКА ПЕРЕДАЧІ ДАНИХ В МЕРЕЖІ

1.1. Теоретичні основи передачі даних в мережі

Розуміння теоретичних основ передачі даних у мережі необхідно для того, щоб зрозуміти, як інформація передається між вузлами.

Моделі можна створювати різними способами в залежності від обраного підходу. Емуляція та моделювання є двома основними підходами, які використовують проміжні методи між використанням фізичних тестових станцій та аналізом аналітичних моделей [11].

Щоб досягти точної емуляції іншої машини, необхідно створити велику колекцію апаратного та програмного забезпечення. Це включає емуляцію апаратного забезпечення та програм машини, яка відрізняється від першої.

Під час створення симуляції метою не є точне повторення дій двох різних систем. Замість цього акцент робиться на точному відтворенні моделі конкретної системи при збереженні всіх її ключових параметрів.

Говорячи про комп’ютери, люди зазвичай використовують терміни емуляція та симуляція як синоніми. Ці терміни стосуються повного копіювання машиною двійкового коду. Навпаки, люди зазвичай використовують ці слова по-різному, обговорюючи абстрактні комп’ютерні моделі.

Емулятори просто взаємодіють із фізичним обладнанням і програмами; вони повністю відрізняються від моделей програмного забезпечення, які містяться в симуляторах. Під час створення симуляції програміст використовує лише віртуальне програмне забезпечення для створення моделі конкретного мережевого пристрою чи протоколу. Крім того, емулятори зазвичай працюють на реальному обладнанні, тоді як симулятори працюють на віртуальних машинах.

Емулятори, які використовують технологію, надану ядром операційної системи, можна розділити на дві категорії. Один тип став можливим завдяки

віртуальним машинам, які використовують фізичне обладнання для запуску програмного забезпечення. GNS3 є прикладом такого типу емулятора. Програмне забезпечення для емуляції поділяється на дві категорії. Перший — це програмне забезпечення, яке працює всередині ядра операційної системи для імітації апаратних пристроїв. Цей тип програмного забезпечення включає Core [1] і Minnet [12], які є розширеннями вбудованої в ОС мережевої системи. Вони можуть працювати в різних процесах на одному ПК, щоб скористатися перевагами мережевих функцій ОС. Іншим варіантом програмного забезпечення емуляції є VT-Mininet [19], який виконується в окремому процесі та використовує мережеві функції ОС.

Існує два види симулятора: дискретно-подієвий і реального часу. Симуляції дискретних подій працюють у певному інтервалі часу; вони дозволяють тестувати різні мережеві протоколи в реальному часі. Деякими прикладами симуляторів дискретних подій є Cisco Packet Tracer і Boson що означає, що вони мають менше ускладнень і їх можна легко переносити між комп’ютерами.

Переваги використання систем імітаційного моделювання полягають у їх здатності точно відтворювати обставини реального життя.

Імітаційне моделювання найкраще використовувати під час вивчення внутрішньої роботи системи. Це може допомогти досліднику зрозуміти ширшу картину та процеси, які вже відбуваються всередині змодельованої системи. Це також може допомогти їм знайти найкращий спосіб дій, враховуючи кожен аспект процесу прийняття рішень. Розбиваючи систему на більш дрібні частини, її можна аналізувати з різних кутів і з різних точок зору. Кожна з цих точок зору дозволяє досліднику побачити, як різні частини взаємодіють одна з одною, що призводить до більш цілісного уявлення про проблему. Цей метод набагато кращий, ніж описувати проблему словами чи математикою. До отримання конкретних результатів моделювання часто припиняється передчасно. Визначення відповідного часу для визначення того, що відбувається в системі, вже забезпечує вирішення проблеми. Це

можна зробити за допомогою статистичної обробки результатів або просто визначити, коли залучені сторони зрозуміють, що відбувається. Протягом багатьох років багато людей вивчали імітаційне моделювання та прийшли до різних висновків щодо нього. Одна група людей досліджувала, наскільки важко створювати моделі, і прийшла до цікавих результатів. Ці результати були проаналізовані багатьма різними людьми, такими як Ф. Мартін і В. Келтон. Вони виявили, що імітаційне моделювання вимагає від експертів багато часу та енергії. Безсумнівно, цей процес є технологічним, оскільки на його основі створюють експериментальні моделі. Однак це не означає, що кожна модель, створена за допомогою імітаційного моделювання, повинна вважатися правилом.

- Вивчаючи взаємодію між декількома частинами, програми, розроблені з використанням імітаційного моделювання, можна використовувати для дослідження систем та їхніх частин.

NetSim. Симулятор дискретних подій використовує математичну модель комп'ютерної мережі. Це дозволяє автору обчислити, скільки даних передається по мережі та скільки даних передається через кожен пристрій у мережі. Однак неможливо налаштувати окремі пристрої, як у реальних мережах. Цей тип тренажера можна знайти в ns-3 [14].

Оскільки емулятори використовують протокол реального світу, вони надають можливість перевірити функціональність основної мережі комп'ютера. Це також дозволяє їм взаємодіяти з традиційними комп'ютерними мережами та програмами. Емулятори дозволяють користувачам тестувати свої системи в умовах, схожих на реальне життя. Однак це відбувається за рахунок збільшення використання системних ресурсів і зниження продуктивності. Крім того, обмеження віртуальних машин означають, що емулятори корисні лише на певних пристроях.

Симулятори дозволяють створювати сценарії, які важко створити в реальному житті. Це робить їх корисними для тестування нових мережевих протоколів перед впровадженням їх у реальному житті. Вони також

портативні та мають вищу продуктивність, ніж реальні мережі. Це пояснюється тим, що вони не вимагають використання операційної системи, Побудувавши модель певної системи, можна оцінити вплив середовища, інших організацій і даних на загальну продуктивність системи.

- Спостерігаючи за результатами моделювання та коригуючи налаштування, відкриваються нові відомості про важливі змінні. Це призводить до створення кращих проектних пропозицій шляхом включення інформації, отриманої з імітаційної моделі.
- Використання моделювання для покращення рішень, знайдених під час аналізу, є особливістю багатьох аналітичних моделей. Їх також можна використовувати для перевірки аналітичних рошень для забезпечення точності.
- Використання імітаційних моделей може допомогти перевірити нові проекти чи стратегії перед їх впровадженням. Це можна використовувати для прогнозування майбутніх результатів.
- Для визначення необхідних вимог до пристрою чи системи використовується імітаційне моделювання.
- Навчання персоналу складним технологічним процесам не потребує ламання або поломки дорогого обладнання. Замість цього можна використовувати імітаційні моделі, не випускаючи з уваги витрати.
- Ви можете використовувати інструменти для анімації змодельованих процесів у навчальних цілях.
- Щоб зрозуміти сучасне виробництво, його необхідно інтерпретувати за допомогою імітаційних моделей.
- Розуміння роботи Р. Шенона та Дж. Бенкса за допомогою аналізу показує, коли немає необхідності виконувати імітаційне моделювання. Це тому, що певні ситуації цього не вимагають.
- Вирішення проблеми вимагає логічного аналізу ситуації.
- Вирішити проблему можна за допомогою теорії SMO.

- Виконуючи прямі практичні експерименти, не порушуючи поточні технології, такі як система обліку часу на робочому місці, можна отримати результати.
- Немає достатньо ресурсів, щоб вчасно створити симуляцію проекту.
- Важко зібрати необхідні дані для моделювання. Крім того, дані можуть бути недоступними або непридатними для викщристання.
- Менеджери проекту вимагають занадто багато даних від імітаційного моделювання занадто швидко.
- Масштаб і складність поведінки змодельованої системи ускладнює її розуміння. Наприклад, важко зрозуміти поведінку погодної системи.

1.2. Технології передачі даних

Програми емулятора GNS3 дозволяють користувачам запускати моделювання GNS3 в іншому програмному забезпеченні.

GNS3 — це програма з відкритим вихідним кодом, призначена для тестування віртуальних або реальних комп’ютерних мереж. Користуватися ним можна безкоштовно та не потребує підписки.

Універсальне програмне забезпечення GNS3, GNS3-gui, забезпечує графічний інтерфейс користувача. Віртуальна машина GNS3, GNS3 VM, працює на окремій операційній системі.

GNS3-все-в-одному — це компонент інтерфейсу користувача GNS3. Його можна використовувати для налаштування мереж з різними топологіями через графічний інтерфейс, як показано на рис. 1.

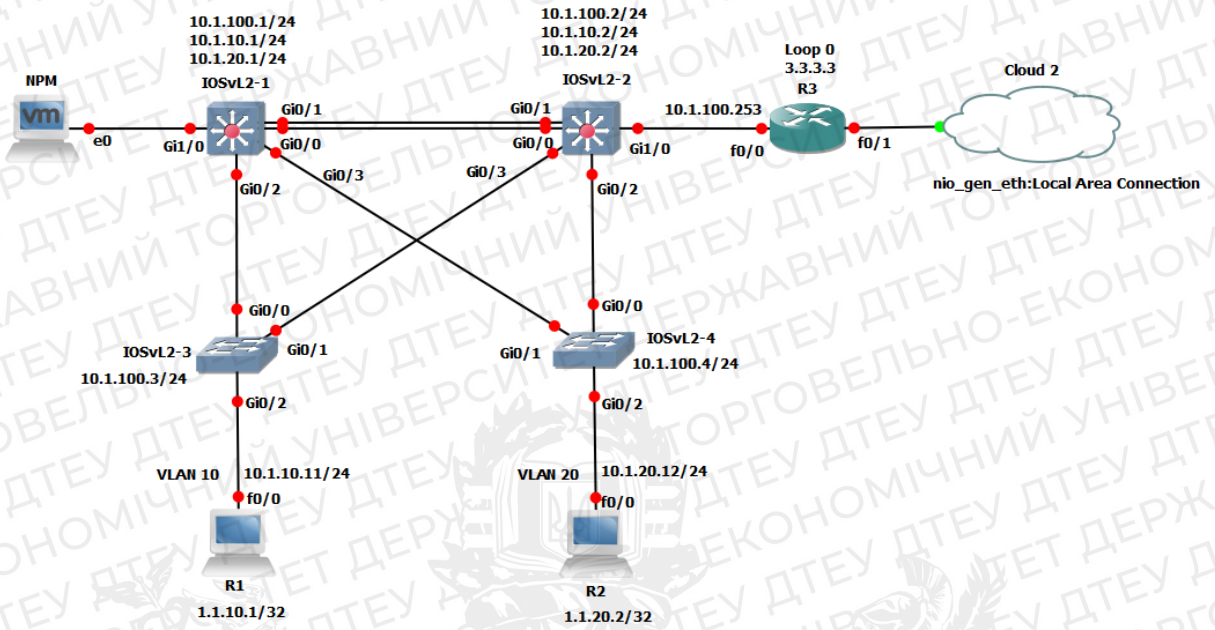

Рис. 1. Мережа зконфігурована в GNS3.

Система GNS3 складається з двох частин: серверної та клієнтської програми. Віртуальну машину можна запускати віддалено або локально на Windows, Linux або програмному забезпеченні VMware Workstation або Virtualbox. Крім того, його можна розгорнути на сервері з можливостями VMware ESXi. Також підтримується розгортання на хмарних серверах.

Віртуальна машина GNS3 підтримує як емуляцію, так і імітацію мережевого обладнання.

Емуляція дозволяє користувачам запускати реальні образи програмного забезпечення мережевих пристроїв за межами їх реальних операційних систем. Наприклад, користувач може скопіювати операційну систему IOS маршрутизатора Cisco та запустити її на емульованому пристрої Cisco GNS3.

Запуск кількох віртуальних машин не є необхідним для моделювання; це економить системні ресурси. Однак можливість запускати реальне програмне забезпечення втрачається, оскільки симуляція виконується всередині віртуальної машини замість того, щоб кожен пристрій мав окремий [6].

Переваги GNS3:

Без оплати. Відкритим вихідним кодом називають програми, написані публічно.

Немає обмежень на кількість дозволених віртуальних пристроїв.

Можливість використовувати програмне забезпечення, вбудоване в реальні мережеві пристрої.

Ця програма підтримує широкий спектр інструментів для віртуалізації різних програм, таких як VMware Workstation, VMware Player і Virtualbox. Крім того, він підтримує ESXi і Fusion.

До недоліків GNS3 можна віднести:

Користувачі повинні самі надати образи програмного забезпечення для пристроїв.

Під час роботи на комп'ютері, розташованому в зоні, необхідні високі вимоги до ресурсів.

Mininet

Створення віртуального контролера, комутатора або маршрутизатора Mininet можливо шляхом підключення до комп'ютерної мережі. Це програмне забезпечення використовує стандартне мережеве програмне забезпечення Linux і підтримує технологію OpenFlow для створення програмно-визначених мереж.

Mininet розгортає мережеві пристрої за допомогою операційної системи Linux і вбудованого програмного забезпечення. Кожен пристрій працює в окремому незалежному процесі для високої продуктивності та масштабування. Система підтримує мережі, які використовують або лише віртуальні пристрої, або реальні пристрої, на додаток до мереж з обома методами включення.

Mininet надає багато переваг порівняно з іншими програмами віртуальної реальності.

Жодних витрат.

Відкритий вихідний код означає загальнодоступне програмне забезпечення, яке кожен може переглядати та редагувати.

Перевершують підходи на основі віртуалізації завдяки підвищеній продуктивності та розміру.

Існують можливості швидкого перезапуску та переналаштування.

Програмне забезпечення, яке можна запустити на реальному обладнанні через мережеву функцію.

Зараз можливе підключення до фізичних мереж.

Mininet має кілька недоліків.

Робота з іншими операційними системами вимагає використання віртуальної машини. Лише Linux офіційно підтримується його хостами.

При великих навантаженнях операційної системи вимірювання параметрів емульованої мережі стають менш точними. Цей недолік виправлено у версії VT-Mininet шляхом додавання підтримки віртуального часу (також називається віртуальним годинником) замість стандартного часу.

Симулятори

Cisco Packet Tracer — це прикладне програмне забезпечення, яке використовується для дослідження пакетних даних.

Додаток PACKET TRACER, розроблений Cisco Systems, є симулятором мережі передачі даних, який дозволяє користувачам створювати робочі моделі мережі, налаштовувати маршрутизатори та комутатори за допомогою команд IOS і взаємодіяти між кількома користувачами через хмару. Комутатори та маршрутизатори Cisco 800, 1800, 1900, 2600, 2800, 2900 і Catalyst 2950, 2960, 3560 доступні через симулятор. На додаток до цього в програму входить брандмауер ASA 5505. Підключення може бути досягнуто за допомогою кабелів, таких як оптоволокно, коаксіальний кабель і послідовний кабель. Телефонні лінії також можуть бути з'єднані попарно для досягнення зв'язку між мережами.

На малюнку 2 представлено зразок формування мережі Packet Tracer.

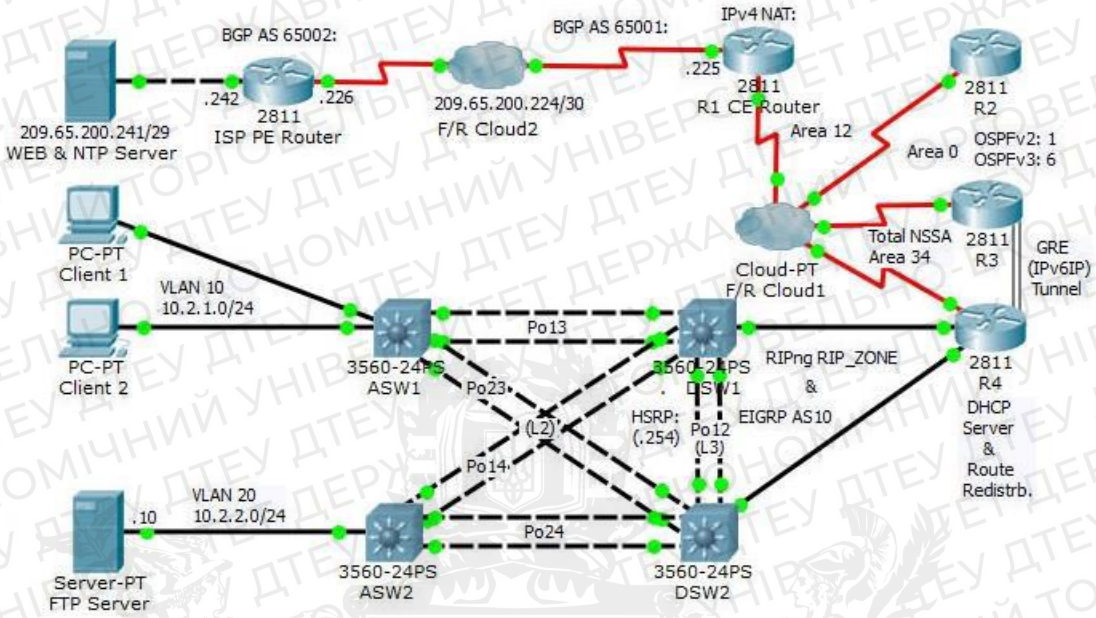

Рис. 2. Мережа, налаштована в Packet Tracer

Переваги Packet Tracer:

- дозволяє створювати та перевіряти працездатність складних мережевих топологій;

– підтримує симуляцію найпоширеніших мережевих пристроїв Cisco;

– працює на комп'ютерах під керуванням ОС Windows та Linux, а також на мобільних пристроях з iOS та Android (Packet Tracer Mobile);

- Має невисокі системні вимоги.

Недоліки Packet Tracer:

– доступний безкоштовно лише зареєстрованим учасникам та інструкторам програми Cisco Networking Academy.

ns-3

ns-3 - дискретно-подійний мережевий симулятор, націлений на використання в дослідницьких цілях і для навчання. ns-3 спроектований таким чином, щоб покривати весь процес симуляції від завдання конфігурації до збору статистичних даних та їх аналізу.

ns-3 підтримує симуляцію як мереж IP, так і мереж, що не використовують протокол IP. Також забезпечена підтримка таких технологій,

як Wi-Fi, WiMAX та LTE, та різноманітні протоколи статичної та динамічної маршрутизації, такі як OLSR та AODV [14].

Переваги ns-3:

- Безкоштовність;
- Відкритий вихідний код;
- Можливість симуляції великих комп'ютерних мереж;
- Можливість обробки трафіку реальних мережевих пристроїв.

Недоліки ns-3:

- Висока складність конфігурації;
- Недоступність на платформі Windows.

1.3 Моделі передачі даних

Системи програмного забезпечення необхідно розробляти та впроваджувати на основі детальної моделі комп’ютерної мережі, з якою вони мають працювати. Цей процес необхідно розпочати до створення програмних систем.

І модель рівня OSI, і стек протоколів TCP/IP використовують еталонні моделі.

Еталонні моделі протоколів OSI та TCP/IP є найпоширенішими. Обидва вони мають справу з властивостями шарів, але модель OSI застаріла і використовується рідко. Його замінено на модель TCP/IP, яка також має більш популярні протоколи.

OSI означає Open Systems Interconnect, яка є еталонною моделлю для комп’ютерів, як показано на малюнку 3. Ця модель має сім рівнів: фізичний,

канальний, мережевий, транспортний, сеансовий і прикладний.

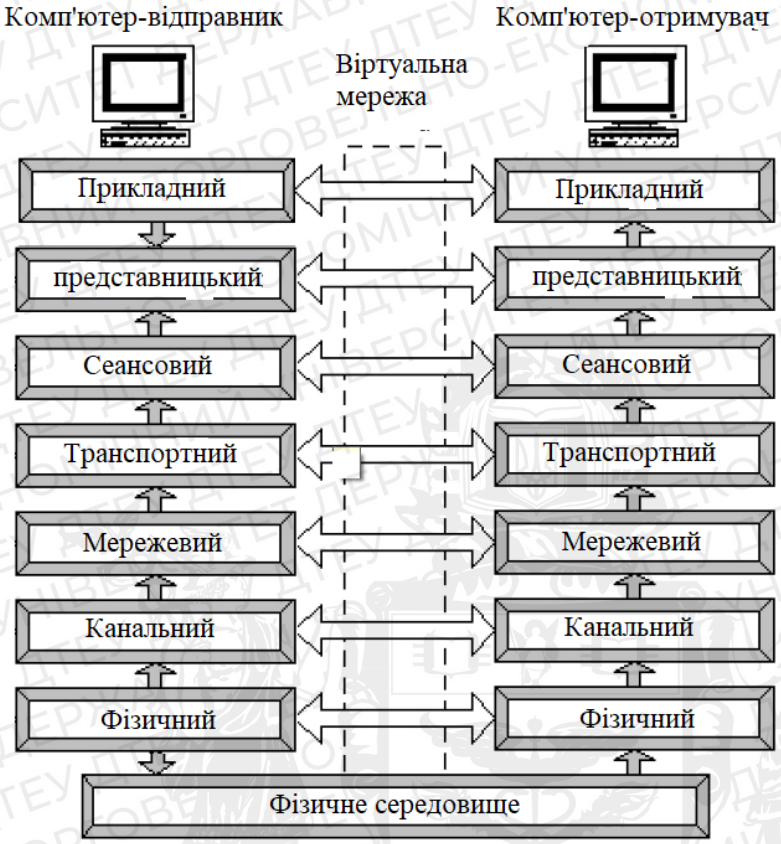

Рис. 3. Схема взаємодії комп’ютерів в еталонній моделі OSI.

Аналогічно моделі OSI, модель TCP/IP має багаторівневу структуру, яка складається з чотирьох рівнів: інтерфейсного, міжмережевого, транспортного та прикладного.

На рис. 4 зображено таблицю порівняння структури рівнів моделей OSI та TCP/IP.

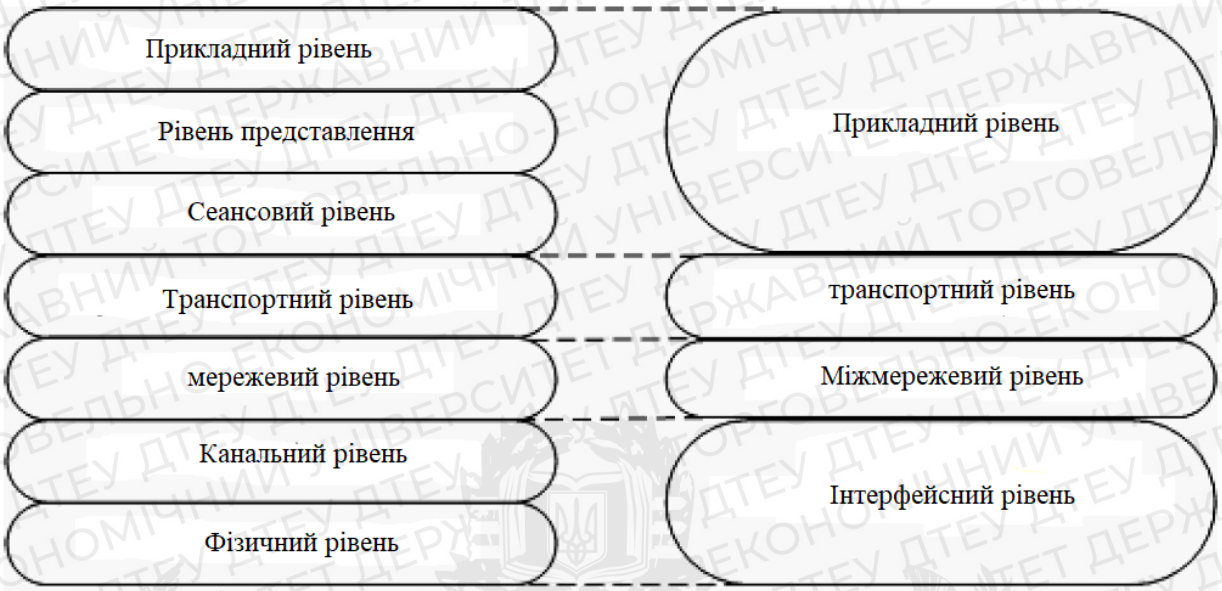

Рис. 4. Порівняння моделей OSI та TCP/IP

Незважаючи на схожість між собою, ці моделі мають багато відмінностей. Три основні концепції моделі OSI відрізняють її від інших моделей: протоколи, інтерфейси та служби. Служба визначає, що робить певний рівень; інтерфейс визначає, як отримати доступ до цього рівня. Деталі внутрішньої реалізації кожного рівня визначаються протоколами, що використовуються цим рівнем. Модель OSI забезпечує кращий розподіл між протоколами, інтерфейсами та послугами, ніж модель TCP/IP. Це тому, що були зроблені спроби змінити оригінальну модель TCP/IP, щоб вона була більш схожою на модель OSI. Однак ця спроба була врешті-решт невдалою, оскільки протоколи важче відокремити в моделі TCP/IP, ніж у моделі OSI. Крім того, змінювати технології набагато легше з багаторівневими протоколами моделі OSI, ніж з моделлю TCP/IP.

Еталонна модель OSI виявляється ефективним представленням абстрактної структури комп'ютерної мережі. Навпаки, модель TCP/IP включає старі протоколи, які використовувалися протягом тривалого часу. Розробники програмного забезпечення при створенні своєї системи вирішили скористатися золотою серединою. Вони вирішили використовувати стекову модель OSI, яка визначає загальну інформацію про мережі, а також протоколи на основі моделі TCP/IP. Через особливі потреби в програмному

забезпеченні необхідний подальший аналіз фізичного, канального та мережевого рівнів моделі.

Фізичний рівень

Кожен пристрій у мережі несе відповідальність за реалізацію фізичного рівня, який стосується передачі даних фізичним каналам зв’язку. Кожну функцію фізичного рівня можна знайти на кожному підключеному пристрої. Інформація на фізичному рівні не має значення для почуттів. Дані повинні надаватися без спотворень або через встановлені інтервали за годинником (інтервал між кожним бітом) [22].

Фізичні носії поділяються на дві основні категорії: дротові та бездротові. І оптичні волокна, і лінії електропередач належать до категорії провідних середовищ. Навпаки, радіохвилі, інфрачервоні промені та видиме світло належать до категорії носіїв бездротового зв’язку. Додаткові приклади включають зв'язок через волоконно-оптичні кабелі, мікрохвильові хвилі та кручені лінії електропередач [22]

З багатьох особливостей, пов’язаних із тим, як трафік передається через фізичні канали, деякі з найважливіших перераховані нижче.

Пропоноване навантаження - це дані, що надходять від користувача до входу в мережу. Це можна виміряти шляхом вимірювання швидкості, з якою біти даних надходять у мережу щосекунди, або кібітів, мебітів тощо.

Швидкість передачі даних – це фактична швидкість, з якою дані передаються через мережу. Це може бути нижчим за швидкість передачі даних, якби вони не були пошкоджені чи втрачені в мережі.

Максимально можлива швидкість передачі інформації по каналу називається пропускною здатністю каналу зв'язку.

При створенні передавача для пристрою зв’язку необхідно враховувати інформацію та особливості середовища фізичної передачі. Крім того, обраний спосіб передачі повинен відповідати бітрейту каналу. Це важливо, оскільки дозволяє передавачу працювати з тією ж швидкістю, що й смуга пропускання каналу.

Дані передаються через абстрактний набір індикаторів замість конкретних носіїв даних. Це допомагає зменшити плутанину, видаляючи будь-які конкретні параметри для підключень. Те, чи працює лінія зв’язку чи передавач із високою швидкістю передачі даних, є ключовим для оцінки загальної якості передачі. Це пояснюється тим, що при виборі цих ключових змінних можливі менші затримки передачі.

Фреймування мережі передбачає вибір топології для встановлення способу підключення комп’ютерів, маршрутизаторів та інших комунікаційних пристроїв. Вибір топології передбачає побудову графа з вузлами, що відповідають комп’ютерам і комунікаційному обладнанню. Ребра графа пов’язують фізичні або інформаційні зв’язки між вузлами. При підключенні кількох комп’ютерів до мережі люди повинні вибрати конфігурацію для своїх з’єднань.

Існують логічні та фізичні структури, пов’язані з мережею. Кожен з них складається з з’єднань між комп’ютерами та комунікаційним обладнанням завдяки кабелям. Вони представлені графом із вузлами, що представляють комп’ютери та кабелі, і ребрами, що з’єднують пари вузлів. Логічні зв'язки формуються, коли реалізовано правильну конфігурацію комунікаційного обладнання.

Різні топології класифікуються як повністю з’єднані або неповністю з’єднані. У повністю підключеній мережі кожен комп’ютер з’єднаний один з одним безпосередньо. Це показано на рис. 5.

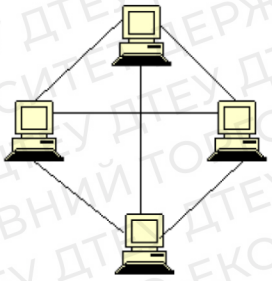

Рис. 5. Повнозв'язна топологія

Всі інші варіанти ґрунтуються на неповнозв'язних топологіях, коли для обміну даними між двома комп'ютерами може знадобитися транзитна

передача даних через інші вузли мережі [21]. На рис. 6 зображено схема топології «зірка» - одного з видів неповнозв'язних топологій.

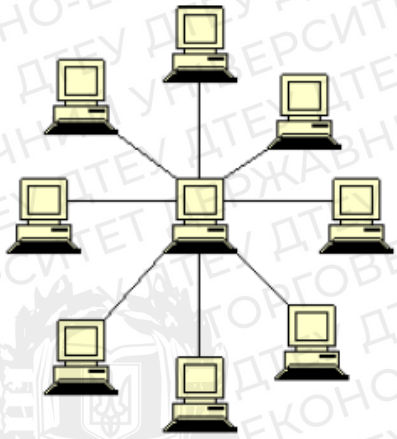

Рис. 6 Топологія «зірка»

Рівень каналу позначається чорним кольором.

Канальні рівні служать зв'язком між мережевим рівнем та іншими рівнями, забезпечуючи прозорий зв'язок між ними. Ці ролі виконуються за допомогою наступних заходів:

З’єднання частин системи під час їх взаємодії.

Необхідність відстежувати швидкість передачі даних між відправником і одержувачем.

Забезпечення надійної передачі, виявлення та виправлення помилок.

Раціональне з'єднання між приладами досягається за рахунок використання прохідного механізму. Це передбачає з’єднання кінцевих частин через проміжні транзитні вузли. Послідовність вузлів на маршруті від відправника до одержувача формує з’єднання. Через один транзитний вузол може проходити кілька маршрутів; йому потрібно знати, як до нього надходять потоки даних, щоб він міг забезпечити правильну передачу кожного елемента до свого інтерфейсу, який підключається до одержувача.

Інформаційні потоки — також відомі як потоки даних — це послідовності даних, які мають подібні характеристики. Це робить їх візуально відмінними від іншого мережевого трафіку. Мітки даних мають глобальне або локальне значення. Прикладом функції пристрою, яка визначає

потік потоку даних, є час надходження на певний інтерфейс. Іншими прикладами функцій, які діють локально всередині пристрою, є глобальні атрибути потоку, такі як адреси кінцевих точок [21].

Комутація пакетів є одним із методів вирішення проблеми комутації. Це вимагає постійного з'єднання між передавальним і приймальним вузлами. Це гарантує, що дані не можуть передавати маршрути зі спільними ділянками. Наразі цей підхід не працює.

Підмережі даних випливають із джерела та звужуються в місці призначення. Пакети виходять із джерела, а потім рекомбінуються під час перемикання між ними. Результатом цього процесу є кілька кадрів, кожен із яких складається із заголовка та поля даних, сформованих із переданих пакетів. Пакети даних розміщуються в середині кількох кадрів. Цей підхід дозволяє передавати дані через одну ділянку мережі без створення прямого з’єднання. Цей метод дозволяє проходити декілька інформаційних потоків через одну фізичну область мережі [21].

Забезпечує підключення через кабель Ethernet.

Стандартно кабелі Ethernet і електричні сигнали знаходяться на фізичному рівні. Ці електричні сигнали контролюються протоколами керування доступом до середовища та форматами кадрів на рівні каналу. В даний час Ethernet є найпоширенішим типом комп'ютерної мережі в світі.

Розмір зображення та формат кадру є ключовими компонентами зображення.

Кадри даних позначені на рис. 7 їх форматом кадру. Вони починаються з 8-байтової преамбули, яка містить послідовність 10101010 і закінчується

роздільником початку кадру. Останній байт кадру є роздільником кінця кадру.

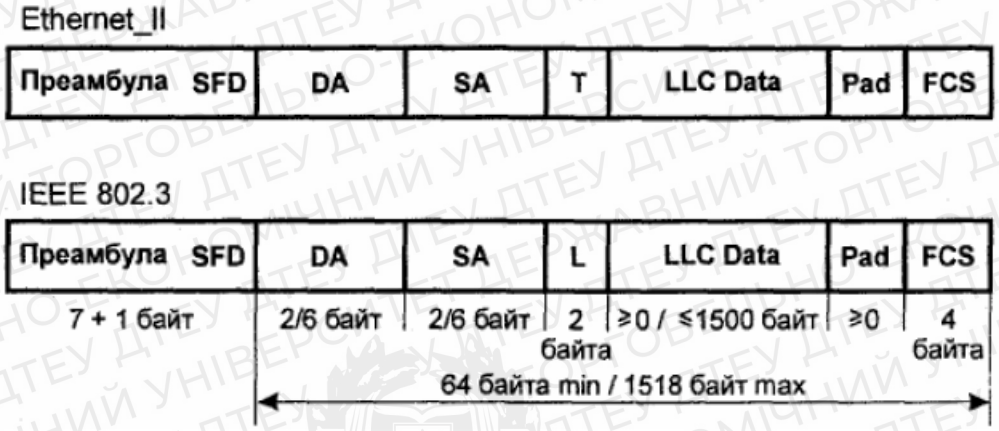

Рис. 7. Формат кадру Ethernet

Додайте ще два поля в кінці повідомлення: одержувач і відправник. Кожен містить шість байтів даних. Під час створення стандарту Ethernet у кожну мережеву карту було обов'язково записувати шестибайтове число, відоме як MAC-адреса. Це служить ідентифікатором як для відправника, так і для одержувача інформації, яка передається через кадр.

Далі йде поле типу кадру, яке надає інформацію про тип даних, що передаються у кадрі. У це поле можна включити дані будь-якого розміру, якщо вони не перевищують 1500 байт.

Останнє поле кадру, контрольна сума, містить 32-бітний код CRC, який виконує виявлення помилок. Це останній аспект стандарту Ethernet, встановлений організацією, яка його створила.

Перемикання

У мережах Ethernet використовується комутація пакетів, яка передбачає використання спеціальних пристроїв, які називаються комутаторами, які передають кадри лише на ті порти, для яких вони призначені. Ці комутатори автоматично перемикають дані між кожним портом. Спеціальні таблиці дозволяють комутаторам перевіряти MAC-адреси вхідних кадрів і визначати, до якого порту їм слід підключитися. Це робиться для маршрутизації кадрів від одного порту до іншого, причому кожен порт комутатора має два з'єднувальні канали до двох різних комутаційних станцій. Ці таблиці

використовуються кожним портом комутатора, оскільки кожен з них перевіряє MAC-адреси вхідних кадрів. Як тільки комутатор знаходить відповідну пару, він пересилає кадр через сполучну лінію між портом одержувача та призначеною станцією одержувача [22].

Використовуючи алгоритм прозорого мосту, міст будує свою таблицю комутації. Він пасивно спостерігає за трафіком, що циркулює в сегментах, підключених до його портів, щоб створити свою таблицю адрес. Приймаючи кадри через свої порти, міст розрізняє, до якого з'єднаних сегментів належить кожен кадр. На основі того, який сегмент вказує адреса джерела, міст визначає, до якої мережі належить вихідний вузол.

Порти комутатора можуть зберігати кадри одним із двох способів. По-перше, якщо будь-які кадри надходять з іншою адресою призначення, вони зберігаються на деякий час у буферній пам'яті. По-друге, порти працюють як кінцеві вузли для своїх сегментів, зберігаючи всі кадри, що надходять. Проте один порт може мати унікальну MAC-адресу, яка не підключена до жодного іншого порту. Це пояснюється тим, що він працює в режимі нерозбірливого захоплення, коли всі вхідні кадри зберігаються протягом деякого часу.[21]

Додаткові наслідки існують без петель, наприклад, відсутність підтримки зациклення мережі. Цей факт ускладнює роботу цих комутаторів.

Рамка збільшується, з'являючись багато разів.

Безперервна циркуляція непотрібних копій кадру в циклі, що заважає мережі нормально функціонувати через її переповнений стан.

Постійне перековування мостів вимагає частих капітальних ремонтів адресної таблиці.

Щоб заблокувати надлишкові з'єднання, комутатори потрібно вручну налаштувати на деактивацію своїх портів. Це тому, що прості мережі не вимагають автоматичного блокування надлишкових з'єднань. Щоб автоматично виявляти цикли, алгоритми використовують підхід охоплюючого дерева (STA). Це найпопулярніший алгоритм, який

використовується у великих мережах з великою кількістю з’єднань — він називається алгоритмом RSTP і STP [7, с. 137].

Завдяки об’єднанню ліній зв’язку підхід резервного з’єднання значно відрізняється від алгоритму дерева покриття. Це пояснюється тим, що їхні методи дуже по-різному справляються з підключенням до Інтернету.

Протоколи STP і RSTP спільно використовують обмежену кількість ліній, які залишаються активними в разі надзвичайної ситуації. Додаткові лінії залишаються доступними для використання у випадку другої надзвичайної ситуації. Це підвищує загальну надійність мережі, але знижує її продуктивність.

Об'єднання резервних каналів підвищує надійність і продуктивність. Фізичні канали об’єднуються для створення єдиного логічного каналу, який має пропускну здатність комбінованих каналів зв’язку. Це відбувається при об’єднанні кількох каналів в один.

Мережевий рівень знаходиться в нижній частині семирівневої ієрархії моделі OSI.

Маршрути доставки пакетів від відправника до одержувача починаються з мережевого рівня. Дані повинні пройти через кілька транзитних ділянок між маршрутизаторами, перш ніж досягти кінцевого пункту призначення. За цей процес відповідає найнижчий рівень архітектури мережі.

Мережевий рівень повинен зрозуміти конфігурацію всієї мережі та вибрати найбільш ефективний шлях через неї. Це може бути складним завданням, оскільки під час вибору маршрутів необхідно враховувати багато маршрутизаторів і ліній. Крім того, під час вибору маршрутів слід ретельно враховувати навантаження на маршрутизатори та лінії. Якщо можливо, цей вибір має призвести до більш рівномірного навантаження на лінії зв’язку та маршрутизатори в мережах. Нарешті, рівень повинен бути в змозі впоратися з будь-якими проблемами, пов’язаними з відмінностями між мережами [22].

З’єднання мереж, побудованих на різних технологіях, вимагає додаткових методів. Ці методи включені до мережевого рівня.

На рівні мережі функції доступні з легкістю.

Набір пов'язаних протоколів.

Маршрутизатори - це спеціальні пристрої з унікальними функціями.

Маршрутизатори з’єднують мережі через фізичні з’єднання. Вони мають багато мережевих інтерфейсів, які можна порівняти з комп'ютерними портами. Кожен інтерфейс маршрутизатора може з’єднувати одну мережу з іншою. Таким чином, усі інтерфейси маршрутизаторів можна вважати вузлами різних мереж.

Завдання рівня маршрутизації полягає в зборі інформації про топологію мереж і створенні таблиць адресації на основі цих даних. Ці таблиці називаються таблицями маршрутизації, і вони включають послідовність мереж або маршрутизаторів, через які має пройти пакет даних, щоб досягти місця призначення. Це важливий аспект роботи рівня, оскільки він впливає на те, як пакети переміщуються мережею та який протокол використовується для їх надсилання. Транспортування пакетів через наступну мережу вимагає багаторівневої структури. Мережевий рівень розміщує пакети в полі даних кадру наступної технології мережевого каналу. Заголовок інтерфейсу маршрутизатора посилається на адресу наступного каналу мережі в заголовку заголовка. Наступна мережа доставляє інкапсульований пакет на призначену адресу. Маршрутизатор перехоплює пакет із навколишнього кадру та обробляє його перед тим, як переслати його до наступної мережі. Роблячи це, маршрутизатор функціонує як координатор для всіх підключених мереж, які працюють разом для досягнення єдиної мети [21].

Визначення найкращого маршруту між двома мережевими інтерфейсами може бути складним. Це тому, що визначення маршруту є складним процесом, який зазвичай зводиться до вибору одного маршруту, який відповідає певним критеріям. Критерії, які використовуються для

визначення оптимальності маршруту, називають метрикою. Для вимірювання довжини маршруту можна використовувати метрику. До них відносяться грошова оцінка, довжина проміжних транзитних вузлів і кількість ділянок. Інші методи вимірювання включають обчислення смуги пропускання кожної ділянки каналу та вимірювання лінійної відстані між двома точками. Зазвичай це робиться шляхом обчислення зворотного значення поточного значення пропускної здатності каналу. Оскільки поточна топологія та склад мережі можуть змінюватися через збій вузлів або нових проміжних вузлів, призначення маршрутів постійно оновлює маршрути та таблиці відповідно до показників. Кожне оновлення також визначає нові маршрути або змінює старі на основі поточних показників [21].

Система вимагає ретельного аналізу передбачуваної тематики та перегляду аналогів. Це призводить до рішення використовувати живу емуляцію мережевого обладнання в реальному часі для системи. Реалізована за допомогою цього методу емуляція обладнання точно повторює мережеві протоколи та стандарти, використовуючи реальні дані. Щоб створити симуляцію, потрібно багато комп'ютерних ресурсів. Це означає високопродуктивні системи з високою абстракцією. Створивши систему кросплатформену, ви зможете уникнути проблем з операційною системою.

При проектуванні системи було вирішено використовувати перші три рівні моделі OSI. Це передбачало використання протоколів зі стеку TCP / IP під час впровадження. Що стосується фізичного та канального рівнів, Ethernet використовується з підтримкою віртуальних локальних мереж (VLAN) і RSTP на канальному рівні. Цей протокол використовується для вирішення проблем із конфігурацією циклу на рівні каналу.

Інтернет-протокол використовує протокол ARP для перетворення IP-адрес у фізичні адреси стандарту Ethernet. Інформація, що надсилається між пристроями, здійснюється за допомогою протоколу ICMP, який маршрутизує маршрут відповідно до алгоритму вектора відстані RIP. Мережевий рівень

реалізований за протоколом IP, а також використовує компоненти вищезазначених протоколів для передачі даних.

2. РОЗРОБКА МОДЕЛІ ПЕРЕДАЧІ ДАНИХ В МЕРЕЖІ

2.1. Проектування схеми мережі передачі даних

Аналіз предмета та існуючих рішень привів до розробки моделі комп'ютерної мережі. Це призвело до формування кількох вимог внаслідок перегляду та створення моделі.

Функціональні вимоги використовують необхідну безперечну функціональність.

Для належної імітації комп'ютерної мережі програмне забезпечення має містити додаткові функції.

Комп'ютери, концентратори, маршрутизатори та комутатори можуть бути створені з різних мережевих топологій.

Коректне використання інтерфейсу командного рядка досягається належним налаштуванням мережевих пристроїв.

Перелічіть діагностичну інформацію про мережеве обладнання, таблиці маршрутизації та список інтерфейсів одним поглядом.

Користувачі використовують команду traceroute і ping у своїх процедурах.

Спочатку нефункціональні вимоги.

Система мала відповідати таким нефункціональним вимогам:

Додаток повинен використовувати принцип симуляції в реальному часі.

Програми повинні використовувати існуючі мережеві протоколи, такі як трансляція адрес ARP, STP spanning tree і алгоритми маршрутизації RIP, а також ICMP.

Щоб конкурувати, проект програмного забезпечення має бути розроблено з використанням мови програмування C# і використовувати платформу Microsoft .NET.

Список вимог визначає, які параметри створюються для системи. Це можна побачити на малюнку 8.

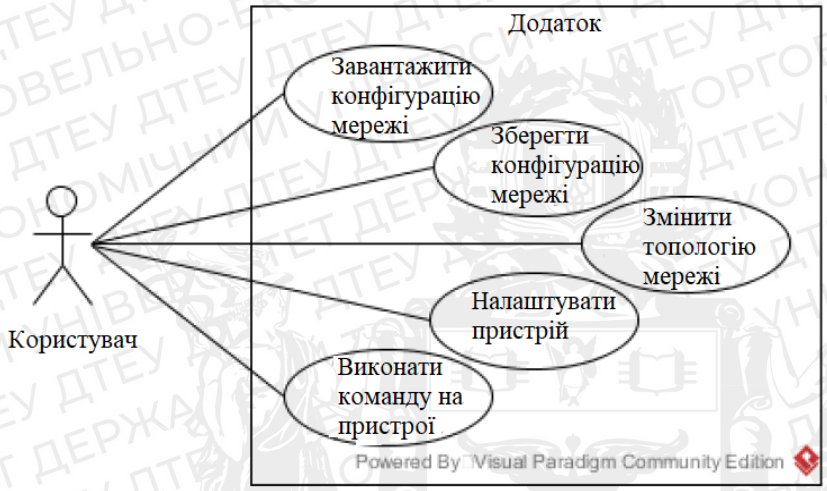

Рис. 8. Діаграма варіантів використання

Параметри мережі для кожного пристрою доступні через файл конфігурації. У цьому файлі детально описана поточна топологія мережі, а також окремі налаштування для кожного пристрою.

Нові пристрої та лінії зв'язку користувач може додавати в мережу.

Користувач може змінювати параметри пристроїв через командний інтерфейс.

Користувачі можуть використовувати програму для діагностики пристроїв і перевірки їх загального стану чи справності окремого пристрою. Крім того, вони можуть використовувати додаток для виконання команд на пристроях.

Спосіб розташування та організації комп’ютерів у мережі.

Необхідно розуміти мережеву структуру комп’ютера, а також основні параметри для кожного пристрою для подальшого розвитку класу.

Мережеве обладнання програми буває двох різновидів: вузли та лінії. Кожен клас цього обладнання представлено окремим обладнанням.

Програма підтримує такі мережеві пристрої через підключення до мережі:

комп'ютер - це кінцевий результат процесу.

Цілеспрямована людина, зазвичай хтось із високою повагою до предмета своїх інтересів.

Пристрій, який перемикає передачі.

Пристрій, який направляє дані, називається маршрутизатором.

Інтерфейс командного рядка дозволяє індивідуально конфігурувати мережеві пристрої, коли користувач правильно налаштовує фізичну структуру мережі. Цей процес називається створенням топології та представлений списком ліній і пристроїв.

Кожен тип мережевого пристрою використовує однаковий набір параметрів, який не залежить від його конкретної природи. Ці параметри наведені в таблиці 1.Табл. 1. Загальні параметри пристроїв

Параметр	Опис
Name	Назва пристрою
Type	Тип пристрою
MACAddress	Фізична адреса пристрою
Interfaces	Набір мережевих інтерфейсів пристрою

Параметр Name є ідентифікатором пристрою, його значення має бути унікальним у межах мережі.

Параметр Interfaces визначає набір мережевих інтерфейсів пристрою. Кількість мережевих інтерфейсів відповідає кількості можливих з'єднань з іншими вузлами мережі.

Для кінцевих пристроїв визначається набір додаткових параметрів, наведених у табл. 2.

Табл. 2. Параметри кінцевих пристроїв

Параметр	Опис
IPAddress	IP-адреса пристрою

IPMask	Маска підмережі
Gateway	Адреса маршрутизатора за замовчуванням

Мережевий інтерфейс є точками підключення ліній зв'язку до пристроїв. Як і пристрої, інтерфейси можуть бути індивідуально налаштовані через інтерфейс командного рядка. Параметри інтерфейсів описані в табл. 3.

Табл. 3. Параметри мережевих інтерфейсів

Параметр	Опис
Name	Назва інтерфейса
MACAddress	Фізична адреса пристрою
IPAddress	IP-адреса пристрою
IPMask	Маска підмережі
VLANTag	Ідентифікатор VLAN
Bandwidth	Пропускна спроможність інтерфейсу, Мбіт/с

Параметр Name є ідентифікатором інтерфейсу, його значення має бути унікальним у межах одного пристрою.

Параметри MACAddress, IPAddress та IPMask є необов'язковими, їх наявність залежить від типу мережного пристрою. Якщо MAC-адреса інтерфейсу не вказана, то як неї використовується фізична адреса пристрою.

Параметр VLANTag призначається лише інтерфейсам комутаторів і позначає ідентифікатор віртуальної локальної мережі, який буде вказано у вхідних кадрах, якщо інтерфейс працює як access.

Параметр Bandwidth означає пропускну спроможність мережного інтерфейсу, тобто максимальний обсяг даних, що проходять через інтерфейс в одиницю часу.

Лінії зв'язку служать підключення мережевих пристроїв друг до друга і виконують передачу інформації з-поміж них. Параметри ліній зв'язку описані в табл. 4.

Параметри ConnectionA та ConnectionB визначають, які пристрої з'єднуються та за допомогою яких інтерфейсів. Є парою ідентифікаторів пристрою і мережевого інтерфейсу.

Табл. 4. Параметри з'єднань

Параметр	Опис
ConnectionA, ConnectionB	З'єднані пристрої
Bandwidth	Пропускна спроможність інтерфейсу, Мбіт/с
Latency	Затримка передачі данних, мс

Параметр Bandwidth позначає пропускну здатність лінії зв'язку, тобто максимальний обсяг інформації, що передається по ній, в одиницю часу.

Параметр Latency означає затримку передачі даних. Зміна цього параметра дозволяє імітувати довжину ліній зв'язку.

2.2. Специфіка програмної реалізації

Перш ніж приступити до реалізації програми необхідно описати його модульну структуру. Основні компоненти додатка зображені на рис. 9.

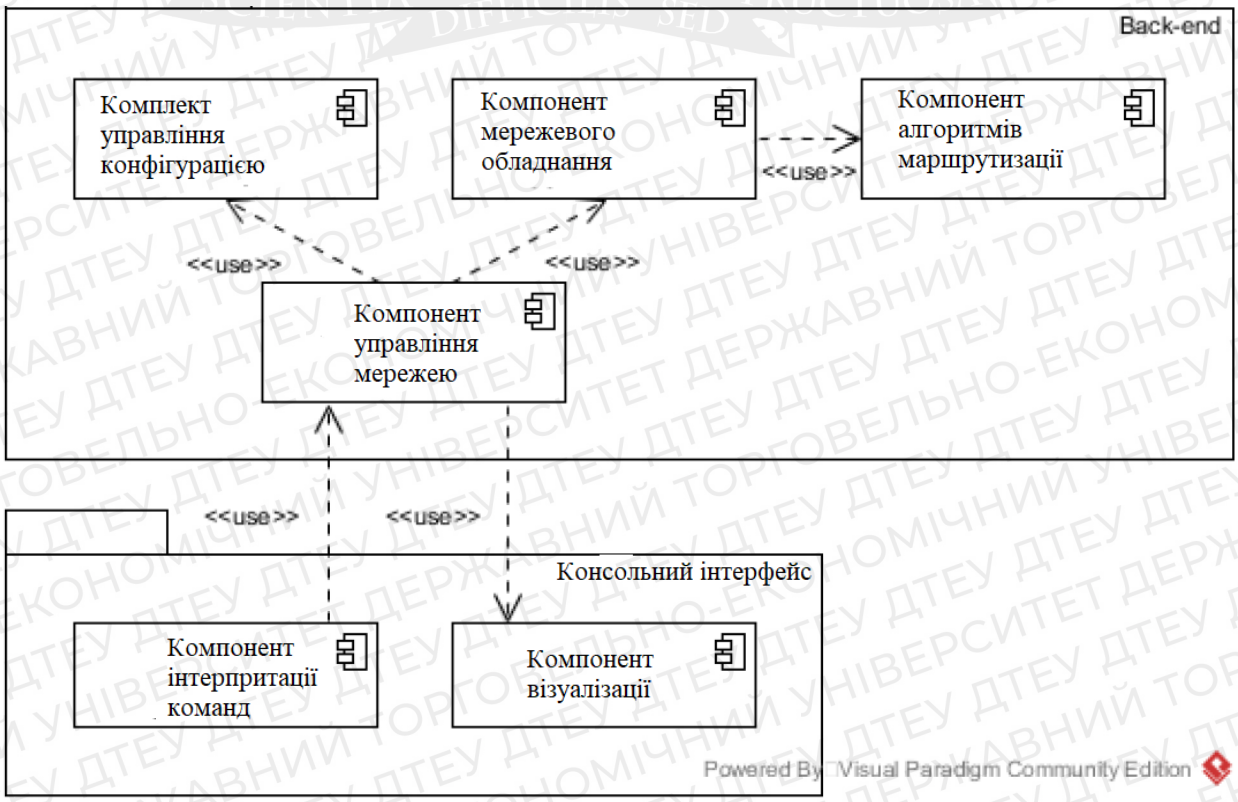

Рис.9. Діаграма компонентів

Інтерфейс консолі представляє модель комп’ютерної мережі, а логічний модуль реалізує протоколи в програмі. Додаткові частини програми складаються з логічного модуля та консольного інтерфейсу.

Компонент керування мережею зберігає поточну топологію мережі, додає або видаляє мережеві пристрої та організовує зв’язок між підключеними пристроями. Цей компонент також забезпечує виконання команд, поданих користувачами.

Працюючи з конфігураційними файлами, компонент керує налаштуваннями в конфігурації мережі комп'ютера. Після завантаження з них конфігураційних файлів компонент зберігає в них свій поточний стан.

Обладнання, що містить протоколи, алгоритми та представлення обладнання, об’єднується як мережевий компонент.

Цей компонент реалізує різні алгоритми створення маршруту. Крім того, це дозволяє додавати додаткові алгоритми під час розробки нових програм.

Коли користувач команди видає команду, компонент інтерпретації команд попередньо обробляє та передає запит компоненту керування мережею.

Компонент візуалізації надає користувачеві інформацію про мережеві пристрої та мережу, а також результати виконаних команд

2.3. Програмне середовище моделювання передачі даних в мережі.

На рис. 10 зображена діаграма послідовності, що демонструє процес взаємодії користувача з системою.

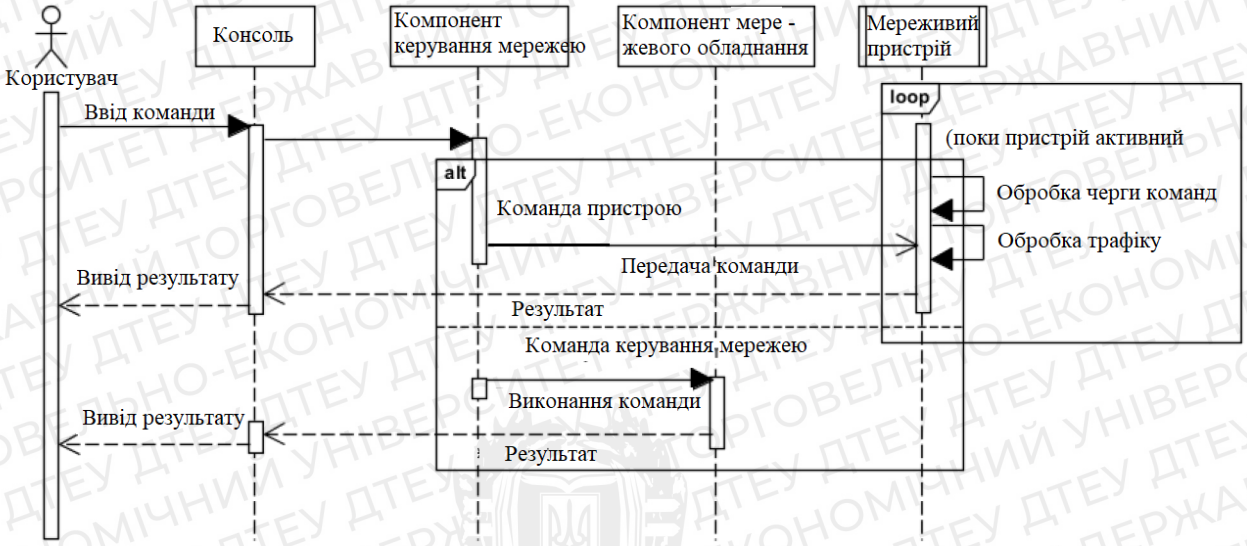

Рис. 10. Діаграма послідовності

Мережеві пристрої безперервно виконують обробку вхідного мережевого трафіку та черги команд користувача. Робота пристрою завершується при отриманні ним команди відключення.

Взаємодія користувача із системою відбувається так:

1) користувач вводить команду в консоль;

2) команда передається на обробку модулю керування мережею;

2.1) якщо команда, що надійшла, адресована пристрою, то вона поміщається в чергу команд пристрою;

2.2) якщо команда, що надійшла, є командою управління то-пологією, то вона передається компоненту мережевого обладнання;

3) результат виконання команди передається компоненту візуалізації і відображається в консолі.

На рис. 11 показаний алгоритм обробки команд користувача і вхідного трафіку мережевим пристроєм.

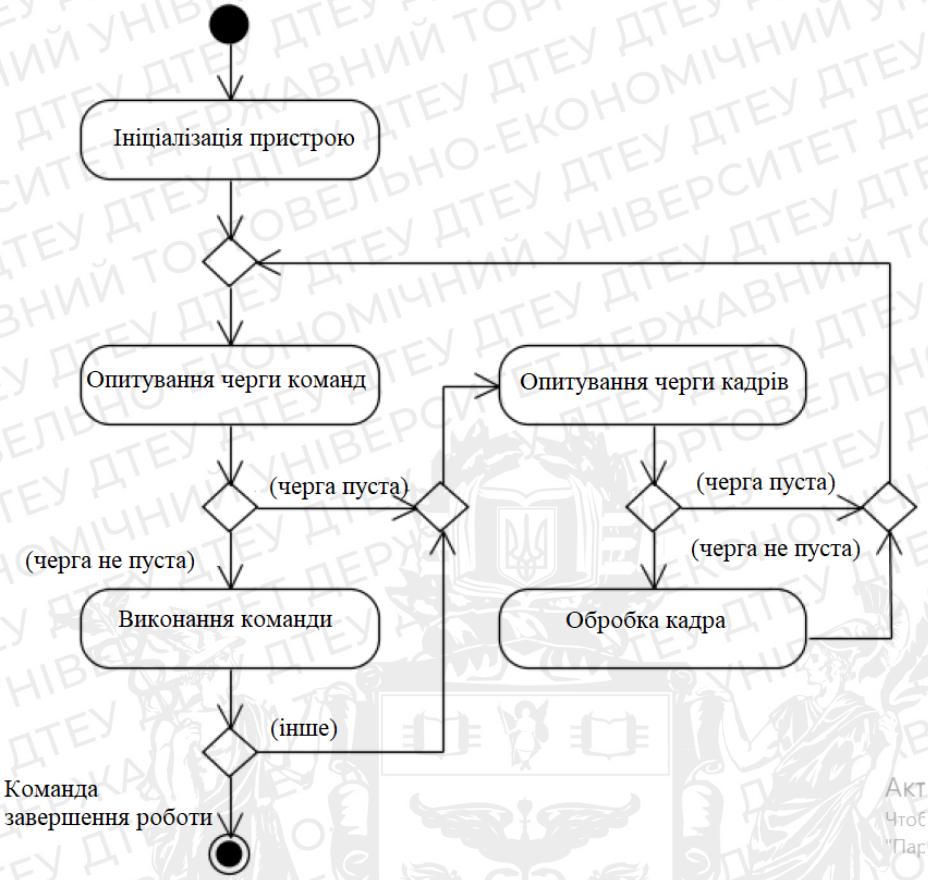

Рис. 11. Алгоритм роботи пристрою

Спочатку пристрій проходить ініціалізацію, під час якої він автоматично підписується на майбутні черги подій і мережеві кадри. Передплатники обробляють кожен кадр або команду до завершення.

Основний процес пристрою продовжується наступними етапами:

Пристрій підтверджує присутність команди в черзі під час перевірки її працездатності.

Коли черга команд не порожня, пристрій вибирає одну команду та обробляє її.

Після обробки команди завершення догляду пристрій припиняє працювати.

Пристрій перевіряє, чи є кадр у черзі кадрів.

Після видалення одного кадру з черги пристрій обробляє його.

Початок циклу переходить з.

Кожен пристрій має власний спосіб обробки команд, враховуючи команди, доступні для цього конкретного пристрою. Крім того, багато загальних команд доступні на всіх пристроях.

Для кожного типу пристрою виконується різна обробка. Це включає керовані комп’ютером відповіді на отримані повідомлення, маршрутизатори, що надсилають кадри призначеному одержувачу, і автоматичні вимикачі, що реагують на коливання напруги.

3. РЕАЛІЗАЦІЯ ІМІТАЦІЙНОЇ МОДЕЛІ ПЕРЕДАЧІ ДАНИХ

3.1. Специфіка побудови імітаційних моделей передачі даних в мережі

Спочатку пристрій проходить ініціалізацію, під час якої він автоматично підписується на майбутні черги подій і мережеві кадри. Передплатники обробляють кожен кадр або команду до завершення.

Основний процес пристрою продовжується наступними етапами:

Пристрій підтверджує присутність команди в черзі під час перевірки її працездатності.

Коли черга команд не порожня, пристрій вибирає одну команду та обробляє її.

Після обробки команди завершення догляду пристрій припиняє працювати.

Пристрій перевіряє, чи є кадр у черзі кадрів.

Після видалення одного кадру з черги пристрій обробляє його.

Початок циклу переходить з.

Кожен пристрій має власний спосіб обробки команд, враховуючи команди, доступні для цього конкретного пристрою. Крім того, багато загальних команд доступні на всіх пристроях.

Для кожного типу пристрою виконується різна обробка. Це включає керовані комп’ютером відповіді на отримані повідомлення, маршрутизатори, що надсилають кадри призначеному одержувачу, і автоматичні вимикачі, що

реагують на коливання напруги.

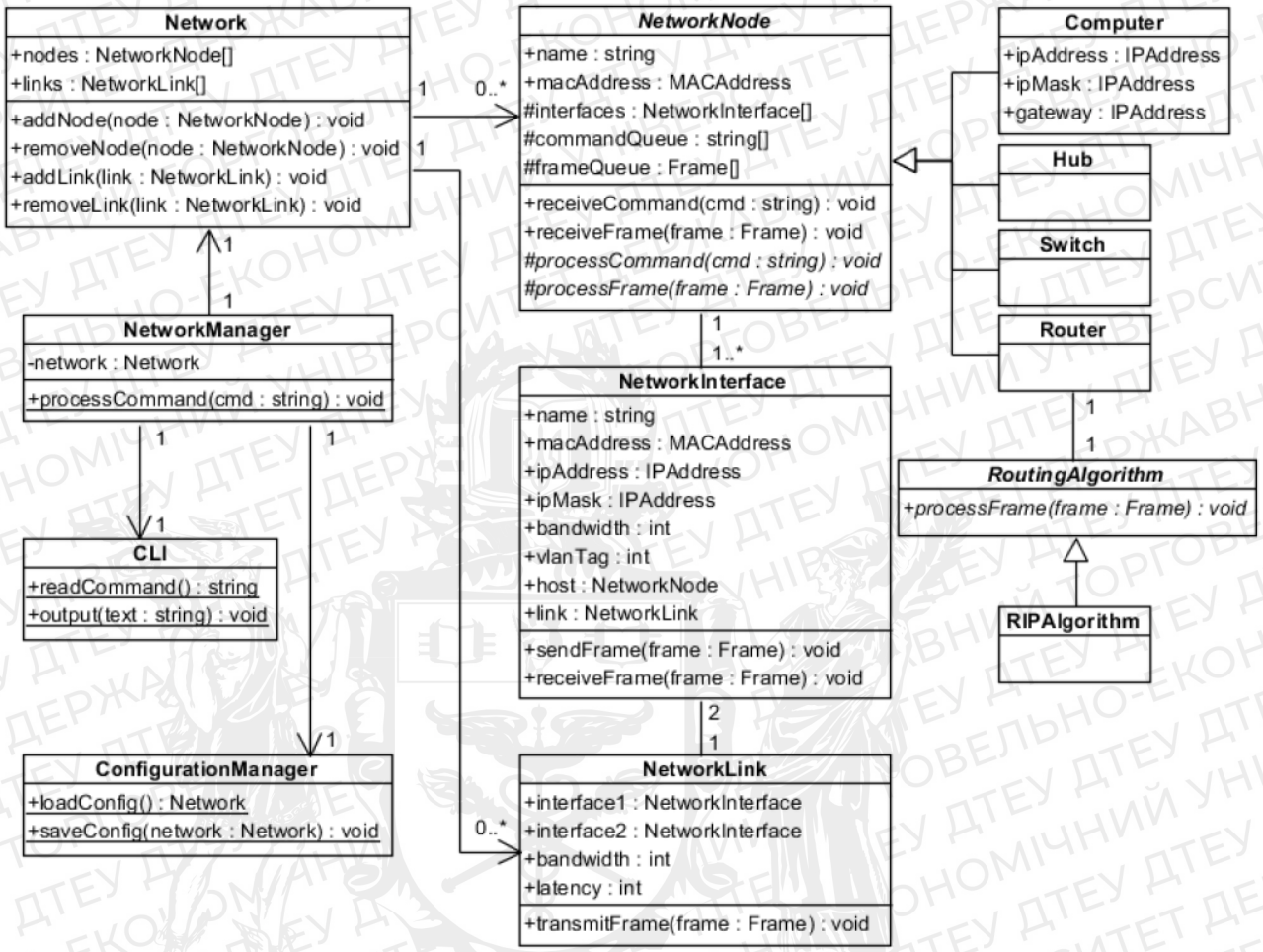

Рис. 12. Діаграма класів

Клас NetworkNode представляє абстрактний мережевий пристрій. Він містить основні параметри, які є загальними для всіх пристроїв класу, наприклад набір мережевих інтерфейсів. Клас реалізує базову логіку для отримання, буферизації та пересилання мережевих кадрів. Абстрактні методи processCommand і processFrame перевизначені підкласами для реалізації різних типів пристроїв з різними наборами команд.

Класи Computer, Hub, Switch і Router є дочірніми класами класу NetworkNode. Щоб представити ці дочірні класи, реалізуйте поведінку комп’ютера, концентратора, комутатора та маршрутизатора шляхом перевизначення методів команд процесу та processFrame.

Кілька підкласів містять абстрактні визначення різних алгоритмів маршрутизації. Ці алгоритми маршрутизації представлені класом RoutingAlgorithm.

Клас RIPAlgorithm використовує алгоритм маршрутизації RIP.

Клас NetworkInterface керує підключенням до мережі в пристроях; він також передає та приймає мережеві кадри через лінії зв'язку пристрою.

Клас NetworkLink представляє лінію зв'язку, яка з'єднує два пристрої. Це також дозволяє передавати дані між ними.

Діаграму, що представляє додавання додаткових класів, можна знайти на рис. 13.

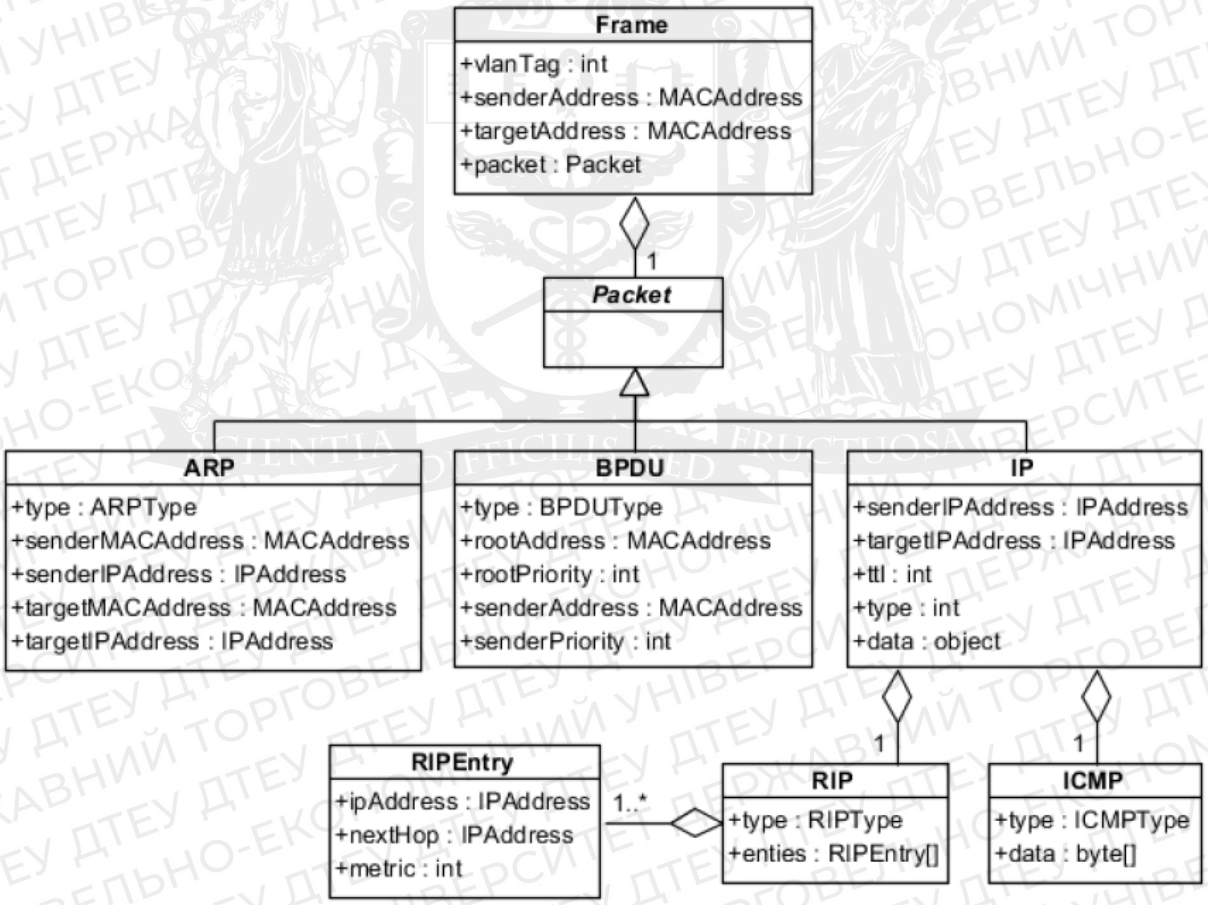

Рис. 13. Допоміжні класи

Об’єкт Frame інкапсулює пакет у своєму кадрі. Його клас, Frame, представляє один кадр у мережі. Він містить фізичні адреси відправника та одержувача, а також ідентифікатор VLAN.

Класи пакетів представляють один пакет, який не є матеріальним. Вони визначають формат абстрактних пакетів різних протоколів.

Протокол ARP визначає структуру кожного свого пакета через клас ARP.

Протокол RSTP використовує певний формат пакету BPDU, визначений класом BPDU.

IP-адреси відправника та одержувача містяться в заголовку кожного IP-пакета. Додатково присутня інформація про довжину пакета, тип даних і поле даних.

Клас протоколу ICMP визначає формат даних, який використовується протоколом ICMP.

Клас RIP визначає формат пакетів, які використовуються для передачі інформації про маршрут RIP.

Клас RIPEntry зберігає інформацію про маршрутизацію пакетів RIP. Він містить IP-адресу наступного маршрутизатора, а також метрику поточного маршруту.

Кожна частина обладнання використовує кілька закодованих процедур для належної роботи та належної взаємодії.

Алгоритми перемикання роботи пристрою.

Коли комутатор визначає вхідні кадри порту, він додає їх неідентифіковане джерело до таблиці комутації. На рис. 14 показаний алгоритм перемикання кадрів.

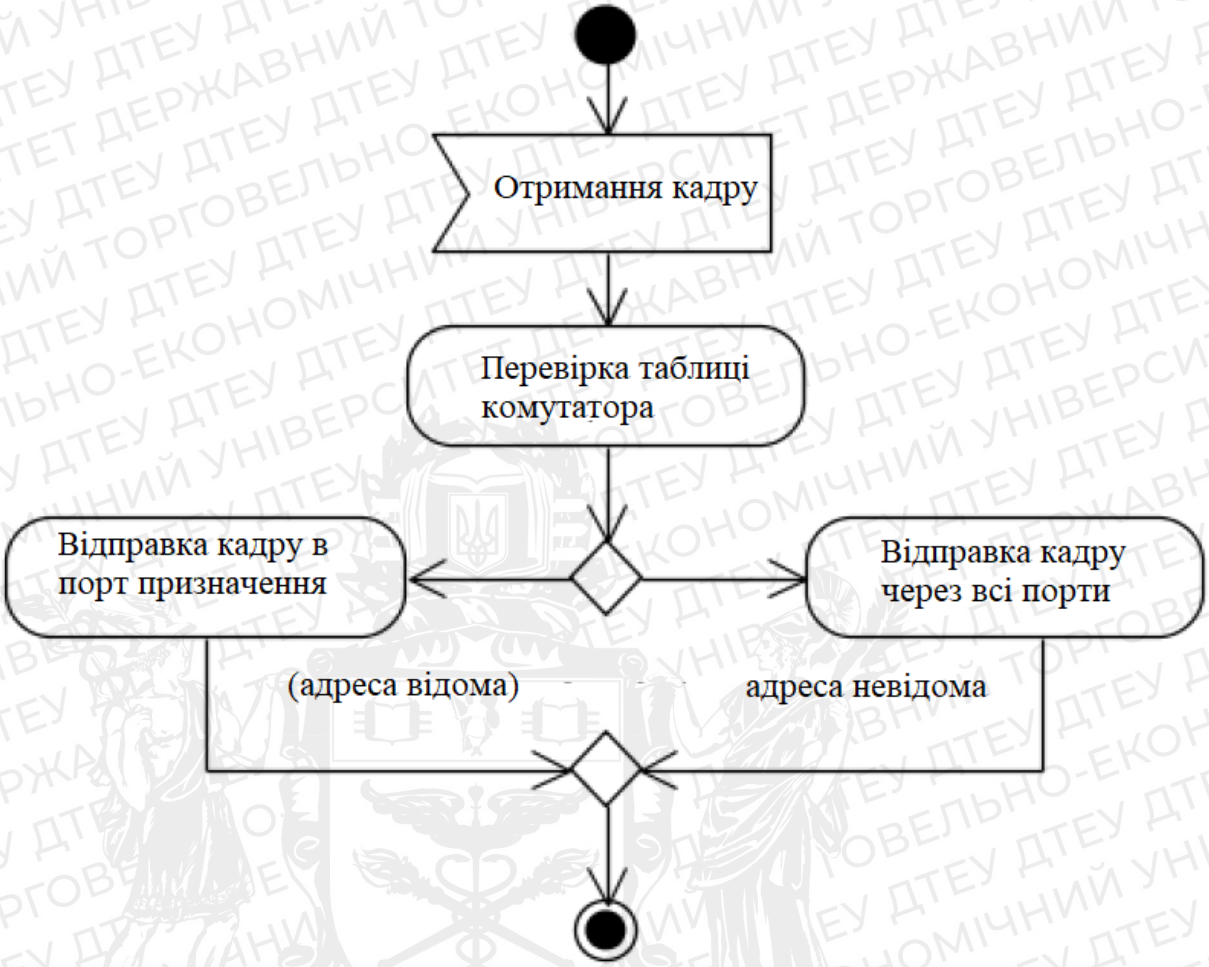

Рис. 14. Алгоритм комутації

При отриманні кадру комутатор звіряє адресу призначення з таблицею комутації. Якщо адреса відома, то комутатор відправляє кадр через відповідну мережевий інтерфейс. В іншому випадку здійснюється розсилка кадру через всі інтерфейси комутатора.

У процесі обробки кадрів відбувається навчання комутатора. При цьому виконується зіставлення MAC-адрес підключених пристроїв з мережевими інтерфейсами комутатора. Алгоритм навчання показаний на рис. 15.

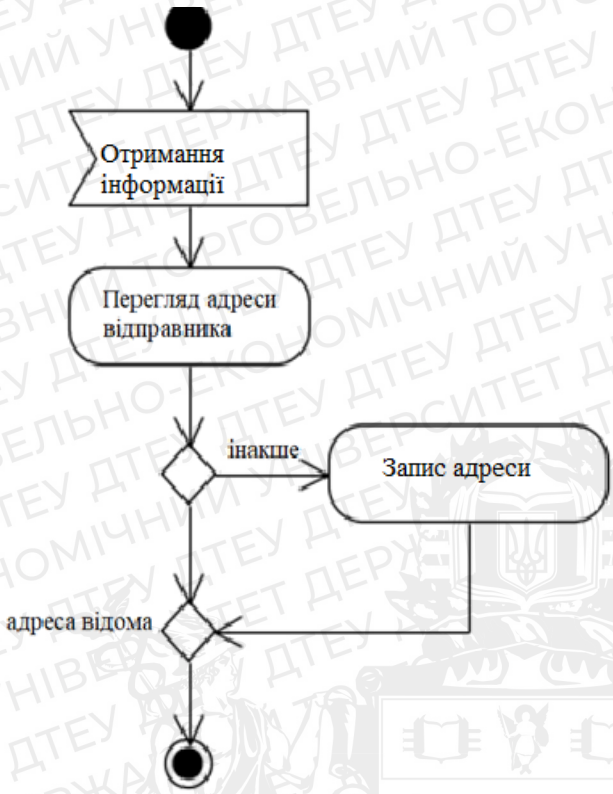

Рис. 15. Алгоритм навчання комутатора

Протокол RSTP використовується для запобігання петлям у топології мережі. Протокол має три основні етапи: виявлення, стабілізація та захист.

Існує єдиний кореневий міст, який визначається випадковим процесом.

Кореневий порт – це порт, пов’язаний з кожним комутатором, який відстежує найкоротший шлях до кореня.

Визначені мости визначають найкоротший шлях перетину між двома сегментами мосту. Одна з секцій мосту, з’єднаних кожним сегментом, стає призначеним портом, який згодом з’єднується з усіма сегментами, які з’єднуються з ним. Якщо міст з’єднує кілька портів з одним портом, у цій ситуації створюється інший призначений порт;

Мости з’єднують усі порти мосту, які не є кореневими чи призначеними портами, до кількох сегментів. Це означає, що всі інші порти заблоковані мостами.

Алгоритм дотримується таких вказівок:

Найдешевшим портом для транспортування даних є кореневий порт.

Визначаючи, який порт використовувати як кореневий, враховуються серійний номер порту та його близькість до інших комутаторів. Додатково враховується пріоритет сусідніх комутаторів.

Вартість портів у грі пропорційна пропускній здатності кожного порту.

Алгоритм RSTP включає наступні процедури:

За замовчуванням кожен комутатор після підключення до мережі вважає себе кореневим.

Кожні 2 секунди комутатори кожного порту надсилають конфігураційний пакет BPDU.

Коли міст отримує ідентифікатор мосту, нижчий за його власний, він припиняє генерацію своїх BPDU і починає повторно передавати BPDU з тим самим ідентифікатором. Таким чином, у мережі залишається лише один міст; він стає кореневим мостом. Крім того, він продовжує генерувати та передавати власні пакети BPDU.

Додавання ідентифікатора об’єднує дані вихідного мосту та збільшує вартість на одиницю. Крім того, інші мости ретранслюють пакети BPDU від кореневого мосту.

Порти, підключені до кількох сегментів міста, визначають призначений порт. Потім через цей порт прибуття пакетів BPDU дані кореневого мосту надходять до кожного підключеного сегмента.

За винятком головного кореневого порту та будь-яких призначених портів, усі сегменти підключаються принаймні до двох портів мосту через заблоковані порти.

Кореневий міст постійно транслює пакети BPDU кожні 2 секунди.

Клас Switch обробляє комутацію та алгоритми RSTP, реалізуючи методи retransmitFrame та processBPDU. Вони також реалізують алгоритм RSTP через processBPDU. Кожен тип набору фреймів аналізується в методі processFrame, а потім обробляється відповідно одним із методів — retransmitFrame або processBPDU.

Алгоритм маршрутизації RIP використовувався для визначення маршруту повідомлення.

Алгоритм вектора відстані RIP можна знайти в класі RIPAlgorithm класу Router, який виконує маршрутизацію.

Для підтримки цілісності розрахунків потрібна окрема система хронометражу.

Коли таймер закінчується, автоматично надсилається оновлення протоколу. Це відбувається через 30 секунд.

Якщо оновлення маршруту не отримано до закінчення 180-секундного таймера, маршрут позначається як недійсний.

Таймер збору сміття дорівнює 240 секундам. Крім того, 60 секунд більше, ніж таймер збирання сміття, є недійсним таймером. Якщо оновлення маршруту надходять до закінчення недійсного таймера, маршрут буде видалено з таблиці маршрутизації.

Формула виглядає так:

Маршрутизатор транслює повідомлення запиту після запуску. Це повідомлення вимагає введення з усіх підключених інтерфейсів.

При отриманні повідомлення-запиту маршрутизатори відповідають відповідним повідомленням, що містить таблицю маршрутизації.

Коли маршрутизатор отримує повідомлення з відповіддю, він дотримується цих правил під час обробки записів маршруту.

У таблицях маршрутизації інформація про маршрути, знайдені в записах маршрутів, з’являється першою.

Якщо нове значення метрики нижче за поточне, таблиця оновлюється.

У випадках, коли нові дані перевищують поточне значення метрики, повідомлення-відповідь має встановити значення метрики на 16, яке вважається нескінченним. Після цього слід використовувати старий маршрут замість нового. Це можна знайти в таблиці зі значеннями показників 3.3.

3.2. Реалізація імітаційної моделі передачі даних в середовищі моделювання Packet Tracer.

У форматі JSON можна зберігати конфігурацію пристрою та дані топології мережі.

У кореневому об’єкті конфігурації зберігаються дві таблиці: table.1 і table. 2. Кожен запис у таблиці 2 є парою, що складається з пристрою та його ліній зв’язку. Масив вузлів у кореневому об’єкті конфігурації містить об’єкти, які описують пристрої, визначені таблицею. 1. Також надається 3 набори мережевих інтерфейсів, які визначаються параметрами, наведеними в таблиці. Таблиця зв'язків містить дані, пов'язані з лініями зв'язку, організовані в таблиці. Масив посилань містить пов’язану інформацію в пов’язаних об’єктах.

Формат MAC-адреси – це рядок, який містить шість однобайтових шістнадцяткових знаків. Ці числа розділяються дефісами. Прикладом MAC-адреси є de-ad-be-af-01-23.

Стандартний запис для IP-адрес комп’ютерів містить чотири однозначні числа, розділені дефісами. Цей формат часто використовується як рядок, як у прикладі вище.

Для визначення ідентифікатора VLAN використовується одне ціле число. Якщо число дорівнює 0, інтерфейс працює в режимі магістралі. І навпаки, додатне значення позначає режим доступу, коли число дорівнює 1.

Розглянемо схему комп’ютерної мережі на малюнку 16. Частина цієї мережі, виділена на малюнку, є сегментом лінії зв’язку між

маршрутизатором і комутатором.

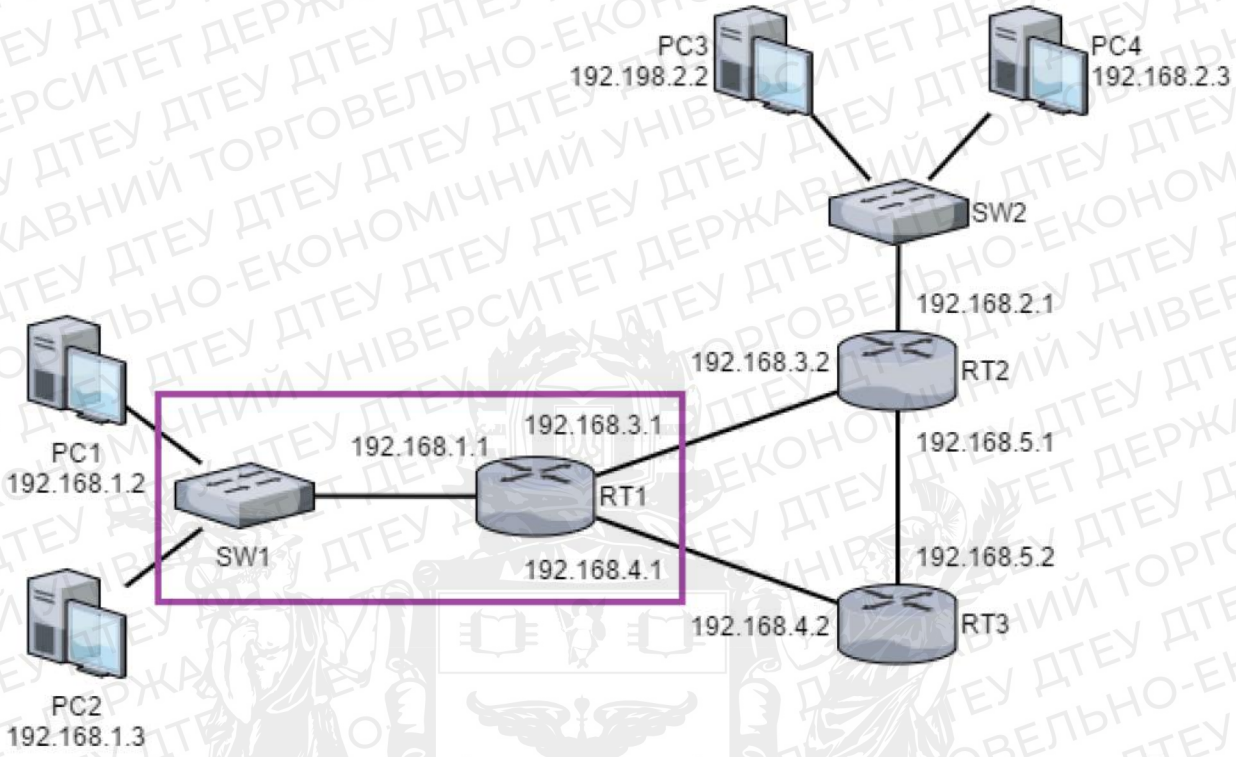

Рис.16. Схема комп'ютерної мережі

Приклад конфігурації комутатора наведено на рис. 17.

```
{
"name": "SW1",
"type": "switch",
"mac-address": "b0-8b-d1-13-a3-9c",
"interfaces": [
{
"name": "port1",
"vlan": 1,
"bandwidth": 100
},
{
"name": "port2",
"vlan": 1,
"bandwidth": 100
},
...
]
}
```

Рис. 17. Конфігурація комутатора

Приклад конфігурації маршрутизатора наведено на рис. 18.

```
{
"name": "RT1",
"type": "router",
"mac-address": "de-ad-be-af-01-23",
"interfaces": [
{
"name": "port1",
"ip-address": "192.168.1.1",
"ip-mask": "255.255.255.0",
```

```
"bandwidth": 100
},
{
"name": "port2",
"ip-address": "192.168.3.1",
"ip-mask": "255.255.255.0",
"bandwidth": 100
},
{
"name": "port3",
"ip-address": "192.168.4.3",
"ip-mask": "255.255.255.0",
"bandwidth": 100
},
...
]
}
```

Рис. 18. Конфігурація маршрутизатора

Приклад конфігурації лінії зв'язку наведено на рис. 19.

```
{
"connection-a": {
"node": "SW1",
"interface": "port1"
},
"connection-b": { 4
"node": "RT1",
"interface": "port1"
},
"bandwidth": 100,
"latency": 5
}
```

Рис. 19. Конфігурація лінії зв'язку

Користувальницький інтерфейс

Для взаємодії користувача з програмою був розроблений набір команд консольного інтерфейсу. Усі команди можна розділити на 3 групи: команди загального призначення, команди редагування мережної топології та команди налаштування мережного обладнання.

Існують три режими роботи командного інтерфейсу:

1) основний режим - у цьому режимі користувач може використовувати команди загального призначення і команди редагування топології мережі;

2) режим керування пристроєм – у цьому режимі користувачеві доступні команди керування пристроєм. Для переходу в даний режим необхідно, перебуваючи в основному режимі, ввести назву устрою;

3) режим керування інтерфейсом – у цьому режимі користувачеві доступні команди керування мережним інтерфейсом. Для переходу в даний режим необхідно, перебуваючи в режимі пристрою, ввести назву інтерфейсу.

Команди загального призначення Команди загального призначення надають базові можливості управління додатком. До команд загального призначення входять:

- help - виводить список доступних команд в залежності від поточного режиму командного інтерфейсу;

– exit – завершує роботу програми або переходить у головний командний режим;

– load <filename> – завантажує конфігурацію комп'ютерної мережі із файлу filename;

– save <filename> – зберігає поточну конфігурацію комп'ютерної мережі у файл filename.

Команди редагування топології

Команди редагування топології дозволяють користувачеві додавати та видаляти мережеве обладнання, а також переглядати поточну топологію. Користувачеві доступні такі команди:

– show – відображає поточну топологію мережі;

– create node <type> <name> <n> – створює новий пристрій типу type з ім'ям name та вказаною кількістю мережевих інтерфейсів n;

– delete node <name> – видаляє пристрій із ім'ям name;

– create link <node-a> <port-a> <node-b> <port-b> <bandwidth> <latency> – створює нову лінію зв'язку між пристроями node-a та node-b через інтерфейси port-a та port -b с зазначеними значеннями пропускної здатності bandwidth і затримки latency;

– delete link <id> – видаляє лінію зв'язку із зазначеним ідентифікатором id. Ідентифікатор лінії зв'язку відображається при перегляді поточної топології комп'ютерної мережі;

- <node-name> - вибирає пристрій з ім'ям node-name в якості поточного і переводить командний інтерфейс в режим керування пристроєм.

Команди налаштування мережевого обладнання

Команди налаштування мережного обладнання відповідають за зміну параметрів мережних пристроїв. Ці команди доступні користувачеві в режимах керування пристроєм та інтерфейсом. Набір доступних команд залежить від типу вибраного пристрою.

Загальні команди

До команд, доступних на всіх пристроях, входять:

– set name <name> – визначає ім'я пристрою;

– set mac <mac> – задає MAC-адресу пристрою;

– show arp – відображає ARP-таблицю пристрою;

- <interface-name> - вибирає інтерфейс з ім'ям interface-name як поточний і переводить командний інтерфейс в режим управління інтерфейсом.

Незалежно від типу пристрою в режимі налаштування інтерфейсу доступні такі команди:

- enable - включає інтерфейс;

– disable – вимикає інтерфейс;

– set name <name> – визначає ім'я інтерфейсу;

– set mac <mac> – задає MAC-адресу інтерфейсу.

Комп'ютер

На комп'ютері доступні такі команди:

– ip – відображає поточну IP-адресу пристрою;

– ip <ip> <mask> – задає IP-адресу та маску підмережі пристрою;

– ip gateway <ip> – задає адресу маршрутизатора за замовчуванням;

– ping <ip> – виконує ping-запит до пристрою з IP-адресою ip.

– tracert <ip> – виконує трасування маршруту до пристрою з IP-адресою ip.

Комутатор

На комутаторі є одна додаткова команда show mac, яка відображає таблицю комутації.

На інтерфейсах комутатора доступні такі команди:

– set mode <mode> – задає режим роботи: access чи trunk;

– set vlan <vlan> – вказує ідентифікатор VLAN для access-інтерфейсу.

Маршрутизатор

На маршрутизаторі доступні такі команди:

– show route – відображає таблицю маршрутизації;

– ip route <ip> <mask> <next-hop> – додає новий запис до таблиці маршрутизації;

– ip rip network <ip> – додає адресу підмережі для протоколу маршрутизації RIP.

На інтерфейсах маршрутизатора доступні такі команди:

– ip – відображає поточну IP-адресу;

– ip <ip> <mask> – задає нові IP-адресу та маску підмережі.

Тестування

Система була перевірена за допомогою функціонального тестування.

Функціональне тестування - це тестування програмного забезпечення з метою перевірки реалізованості функціональних вимог, тобто можливості програмного забезпечення в певних умовах вирішувати завдання, необхідні користувачеві.

Для перевірки працездатності програми була розроблена тестова конфігурація комп'ютерної мережі, схема якої зображена на рис. 20. На основі цієї схеми було розроблено файл конфігурації.

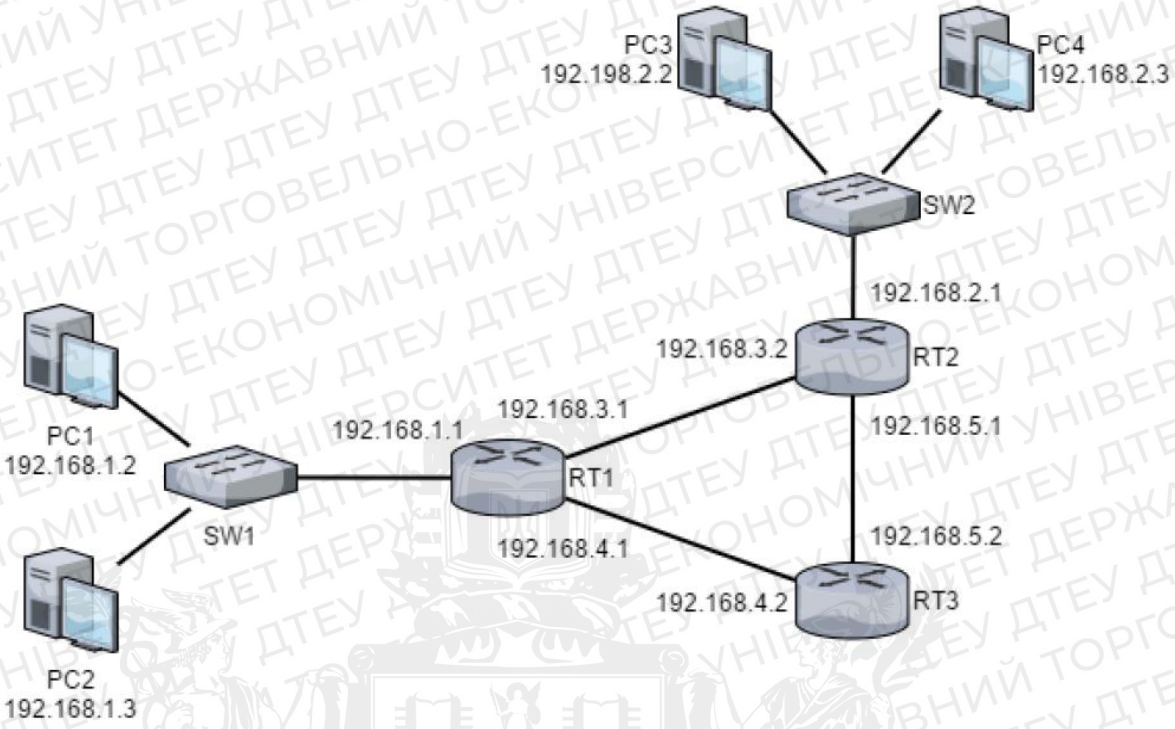

Рис. 20. Схема тестової комп’ютерної мережі

Протокол функціонального тестування наведено у табл. 5.

Табл. 5. Протокол функціонального тестування

№	Назва тесту	Кроки	Очікуваний результат	Тест пройдений?
1	Загрузка конфігурації мережі	1. Користувач вводить команду load із зазначенням файлу конфігурації. 2. Користувач вводить команду show.	Конфігурація комп'ютерної мережі завантажена та коректно відображається на екрані.	Так
2	Перевірка зв’язаності мережі	1. Користувач вибирає пристрій PC1. 2. Користувач вводить команду traceroute	Відображається некоректне трасування маршруту до	Так

		192.168.2.2. 3. Користувач вибирає пристрій RT1. 4. Користувач вводить команду show route.	комп'ютера PC3 і правильна таблиця маршрутизації.	
3	Вимкнення інтерфейсу 192.168.3.1 маршрутизатора RT1 та перевірка зв'язності.	1. Користувач вибирає пристрій RT1. 2. Користувач вибирає інтерфейс port2. 3. Користувач вводить команду dis-able. 4. Користувач вибирає пристрій RT1. 5. Користувач вводить команду show route. 6. Користувач вибирає пристрій PC1. 7. Користувач вводить команду traceroute 192.168.2.2.	Маршрутизатор оновлює таблицю маршрутизації і відновлює зв'язок, використовуючи альтернативний маршрут. Відображаються коректна таблиця маршрутизації та трасування маршруту.	Так
4	Збереження конфігурації	1. Користувач вводить команду save із зазначенням імені конфігураційного файлу.	Успішно створено новий конфігураційний файл.	Так
5	Додавання нового пристрою та перевірка	1. Користувач вводить команду cre-ate node computer PC5 1. 2. Користувач вводить	Пристрій успішно доданий та доступний. Коректно	Так.

	зв'язності.	команду cre-ate link PC5 port1 SW2 port4 100 0. 3. Користувач вибирає пристрій PC5. 4. Користувач вводить команду traceroute 192.168.1.2.	відображається трасування маршруту.	

На рис. 21 та рис. 22 показані результати трасування маршруту від PC1 до PC3 та таблиця маршрутизації RT1 після завантаження конфігурації.

```
PC1> tracert 192.168.2.2

Tracing route to 192.168.2.2:

  1    0 ms      1 ms      0 ms      192.168.1.1
  2    0 ms      1 ms      0 ms      192.168.3.2
  3    1 ms      0 ms      0 ms      192.168.2.2

Trace complete.
```

Рис 21. Трасування маршруту після завантаження конфігурації

```
RT1> show ip

C    192.168.1.0/24 is directly connected, port1
R    192.168.2.0/24 through 192.168.3.2, port2
C    192.168.3.0/24 is directly connected, port2
C    192.168.4.0/24 is directly connected, port3
R    192.168.5.0/24 through 192.168.3.2, port2
                    through 192.168.4.2, port3
```

Рис. 22. Таблиця маршрутизації RT1 після завантаження конфігурації

На рис. 23 та рис. 24 показані результати трасування маршруту від PC1 до PC3 та таблиця маршрутизації RT1 після відключення його інтерфейсу 192.168.3.1.

```
PC1> tracert 192.168.2.2

Tracing route to 192.168.2.2:

  1    0 ms      0 ms      0 ms      192.168.1.1
  2    0 ms      0 ms      0 ms      192.168.4.2
  3    1 ms      0 ms      2 ms      192.168.5.1
  4    1 ms      0 ms      0 ms      192.168.2.2

Trace complete.
```

Рис. 23. Трасуванн маршрута після розриву з’єднання

```
RT1> show ip

C    192.168.1.0/24 is directly connected, port1
R    192.168.2.0/24 through 192.168.4.2, port3
C    192.168.4.0/24 is directly connected, port3
R    192.168.5.0/24 through 192.168.4.2, port3
```

Рис. 24. Таблиця маршрутизації RT1 після розриву з'єднання

На рис. 25 показаний результат трасування маршруту з доданого комп'ютера PC5 до PC1.

```
PC5> tracert 192.168.1.2

Tracing route to 192.168.1.2:

  1    0 ms      0 ms      0 ms      192.168.2.1
  2    0 ms      0 ms      1 ms      192.168.3.1
  3    1 ms      0 ms      1 ms      192.168.1.2
```

Рис. 25. Трасування маршруту від PC5 до PC1

ВИСНОВОК

У ході виконання курсової роботи було реалізовано програму для імітаційного моделювання комп'ютерних мереж.

Основні результати

1. Виконано аналіз предметної галузі та огляд існуючих рішень.

2. Розроблено модель комп'ютерної мережі.

3. Спроектовано програму для імітаційного моделювання комп'ютерної мережі.

4. Реалізовано додаток.

5. Виконано тестування програми.

Напрям подальших досліджень

Подальші дослідження та практичні розробки можуть бути націлені на реалізацію графічного інтерфейсу користувача з візуалізацією роботи мережі, а також на забезпечення можливості розширення шляхом додавання нових мережевих стандартів та протоколів, алгоритмів маршрутизації та мережевих пристроїв.

ЛІТЕРАТУРА

1. Томашевский В.Н., Жданова Е.Г., Жолдаков А.А. Решение практических задач методами компьютерного моделирования: Учеб. Пособие – К.: Изд-во "НАУ", 2001. – 268 с.

2. Ahrenholz J., Danilov C., Henderson T.R., Kim J.H. CORE: A real-time network emulator. // Military Communications Conference, 2008. MIL-COM 2008. IEEE. – P. 1–7.

3. Boson NetSim Network Simulator. [Електронний ресурс] URL: http://www.boson.com/netsim-cisco-network-simulator (дата звернення: 10.03.2017).

4. Cisco Packet Tracer. [Електронний ресурс] URL: https://www.netacad.com/about-networking-academy/packet-tracer/ (дата звернення: 10.03.2017).

5. Git. [Електронний ресурс] URL: https://git-scm.com/ (дата звернення: 15.04.2017).

6. GitHub. [Електронний ресурс] URL: https://github.com/ (дата звернення: 15.04.2017).

7. GNS3 Documentation. [Електронний ресурс] URL: https://docs.gns3.com/ (дата звернення: 10.03.2017).

8. IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. IEEE Std. 802.1D-2004 - IEEE Computer So-ciety, 2004.

9. IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. IEEE Std. 802.1Q-2005 - IEEE Computer Socie-ty, 2005.

10. JSON Specification. [Електронний ресурс] URL: http://www.json.org (дата звернення: 15.04.2017).

11. Kayssi A., El-Haj-Mahmoud A. EmuNET: a real-time network emulator. // Proceedings of the 2004 ACM симпозіум на Applied computing (Nico-sia, Cyprus, 14-17 March 2004), SAC '04, 2004 - P. 357-362.

12. Liu J. A Primer for Real-Time Simulation of Large-Scale Networks. // 41st Annual Simulation Symposium. 2008 - P. 85-94.

13. Mininet Overview. [Електронний ресурс] URL: http://mininet.org/overview/ (дата звернення: 10.03.2017).

14. Newtonsoft Json.NET. [Електронний ресурс] URL: http://www.newtonsoft.com/json (дата звернення: 15.04.2017).

15. ns-3 Overview. [Електронний ресурс] URL: https://www.nsnam.org/overview/what-is-ns-3/ (дата звернення: 10.03.2017).

16. RFC 791 - Internet Protocol. [Електронний ресурс] URL: https://tools.ietf.org/html/rfc791 (дата звернення: 20.03.2017).

17. RFC 792 - Internet Control Message Protocol. [Електронний ресурс] URL: https://tools.ietf.org/html/rfc792 (дата звернення: 20.03.2017).

18. RFC 826 – Ethernet Address Resolution Protocol: Or Converting Network Protocol Address to 48.bit Ethernet Address for Transmission on Ethernet Hardware. [Електронний ресурс] URL: https://tools.ietf.org/html/rfc826 (дата звернення: 20.03.2017).

19. RFC 1058 – Routing Information Protocol. [Електронний ресурс] URL: https://tools.ietf.org/html/rfc1058 (дата звернення: 20.03.2017).

20. Yan G. Improving Large-Scale Network Traffic Simulation with Mul-ti-Resolution Models. Dartmouth Computer Science Technical Report TR2005-558. - Dartmouth College, 2005. - 173 p.

21. Yan J., Jin D. VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation. // SOSR '15 Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Re-search. 2015. - No. 27. – P. 1–7

22. Оліфер В.Г., Оліфер Н.А. Комп'ютерні мережі. Принципи, технології, протоколи. 5-те вид. - СПб.: Пітер, 2016. - 992 с.

23. Таненбаум Е., Уезеролл Д. Комп'ютерні мережі. 5-те вид. - СПб.: Пітер, 2016. - 960 с.

24. Троєлсен Е., Джепікс Ф. Мова програмування C# 6.0 та плат-форма .NET 4.6. 7-е вид. - М.: Вільямс, 2016. - 1440 с.

25. Советов Б.Я., Яковлев С.А. Моделирование систем. - М.: Высш. школа, 2003. - 320с.

26. Советов Б.Я. Яковлев С.А. Моделирование систем: учебник для ВУЗов. – М.: Высш. шк., 1999. – 224с.

27. Ситник В.Ф., Орленко Н.С. Імітаційне моделювання: Навч. посібник. – К.: КНЕУ, 1998. -208с.

28. Шрайбер Т.Дж. Моделирование на GPSS. – М.: Машиностроение, 1980.-593с.

29. Советов Б.Я. Яковлев С.А. Моделирование систем. Курсовое проектирование. – М.: Высш. шк., 1988. – 135 с.

30. Ситник В.Ф., Орленко Н.С. Імітаційне моделювання: Навч.-метод. посібник для самост. вивч. дисц. – К.: КНЕУ, 1999. -208с.

31. Кудрявцев Е.М. GPSS World. Основы имитационного моделирования различных систем. М.: ДМК Пресс, 2004. – 320с.

32. Руководство пользователя по GPSS World. /Перевод с английского/. – Казань: Изд-во «Мастер Лайн», 2002. – 384с.