МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«Мережі операторського класу на основі технології MPLS з використанням протоколу LDP»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Великодний Д.В.**

Студента групи ІН-64-8 — **Некислих О.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____

Зав. кафедрою Довбиш А.С.

«_____» _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-64-8 спеціальності «Інформатика» денної форми навчання Некислих Олени Олексіївни.

Тема: «Мережі операторського класу на основі технології MPLS з використанням протоколу LDP»

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд існуючих рішень; 2) моделювання технології MPLS з використанням симулятора GNS3; 3) розробка графічного інтерфейсу веб-орієнтованої системи.

Дата видачі завдання «_____» _____2020 р.

Керівник випускної роботи _____ Великодний Д.В.

Завдання приняв до виконання _____ Некислих О.О.

РЕФЕРАТ

Записка: 54 сторінки, 23 рисунка, 11 джерел.

Об’єкт дослідження — мережа операторського класу на основі технології MPLS з використанням протоколу LDP.

Мета роботи — розробка веб-інтерфейсу, який дозволить з легкістю налаштовувати на маршрутизаторах конфігурацію інтерфейсів та динамічну маршрутизацію протоколів LDP і OSPF.

Методи дослідження — моделювання схеми мережі за технологією MPLS в симуляторі GNS3. Використання засобів HTML, CSS і JavaScript для розробки графічного інтерфейсу.

Результати — розроблено веб-інтерфейс, який дозволить без зусиль налаштовувати маршрутизатори як в симуляторах, так і на реальному обладнанні. Веб-інтрефейс має вікно, з якого можна скопіювати код налаштувань протоколів OSPF та LDP, і використовувати його в подальшому, що є зручним для користувача.

ВЕБ-ІНТРЕФЕЙС, СИМУЛЯТОР GNS3, ПРОТОКОЛ LDP,
ТЕХНОЛОГІЯ MPLS, ПРОТОКОЛ OSPF.

ЗМІСТ

ВСТУП

В наш час інформаційний зв’язок займає одне з найважливіших місць серед населення. Він весь час розвивається та вдосконалюється, а також потрапляє в усі без винятку області пов’язані з життєдіяльністю людини, в той же час адаптуючись до них. Застосовують зв’язок різноманітні верстви населення та компанії, відштовхуючись від цього всі вони висувають різні вимоги до його якості. Стало бути, щоб забезпечувати необхідні види та рівні обслуговування, сучасним операторам і провайдерам необхідно мати досить гнучку і багатофункціональну мережу яка буде здатна на це. Отже найбільш відомою системою вважається технологія багатопротокольної комутації по мітках (MPLS), яка застосовується на транспортному рівні та також надає можливість гарантувати дуже ефективну передачу даних з підтримкою технології QoS (Quality of Service). MPLS формує основу з метою розгортання нових типів послуг, які не підтримуються традиційною маршрутизацією, в той час вона дає можливість скоротити собівартість та удосконалити якість базових послуг.

В повсякденному житті практично кожен раз стикаємось з тим, що життя неможливе без використання того чи іншого комп’ютеризованого пристрою, тому в такому випадку це призводить до збільшення кількості комп’ютерів і ускладнення взаємозв’язків між ними. З метою спрощення взаємодії комп’ютерів їх пов’язують в мережі. Правильність роботи мережі залежить від того як її налаштувати. У зв’язку з тим, що не всі навчальні заклади мають можливість купувати дороге обладнання провідних фірм, вони виходять з положення тим що застосовують симулятори. Це дозволяє навчитися та застосовувати знання працюючи на найбільш сучасному обладнанні без залучення додаткових ресурсів. За допомогою програми GNS3 спільно з навчально-методичним забезпеченням можна досягти цієї мети, а саме навчитися розробляти свої власні мережі, в яких буде надаватися можливість створювати та відправляти різні пакети даних та зберігати проект.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Технологія MPLS та її складові

Застосування технології MPLS є одним із успішних засобів передачі потоку інформації з параметрами QoS.

MPLS (MultiProtocol Label Switching) — створена в багатопротокольних мережах як технологія високошвидкісної комутації пакетів, яка базується на використанні міток. Для способу побудови швидкісних IP-шляхів розробляється і використовується ця технологія. Застосовується MPLS при поширенні трафіку завгодно якого протоколу маршрутів в мережі і не обходиться IP-протоколом.

Технологія базується на принципі обміну мітками. Пакет який передається ототожнюється певною міткою і має зв’язок з будь-якими класами мережевих рівнів (клас еквівалентності переадресації або FEC). Маршрутизатори, які комутують за допомогою міток (Label Switching маршрутизатор або скорочено LSR) — поля між вузлами-сусідами мереж MPLS та тільки для них вагомість міток є унікальною. Передачі міток відбуваються за допомогою пакетів, які від використання технологій канальних рівнів залежать і від засобів їх прив’язки до пакетів.

Топологічні відомості з мереж, яку будуть отримувати маршрутизатори LSR, які побудувавши зв’язок з роботою деяких алгоритмів маршрутизації, наприклад, OSPF, BGP, IS-IS. Надалі вони приступають до співпраці з деякими маршрутизаторами-сусідами розподіливши мітки, що потім використовуються для комутації. Побудова в середині доменів шляхів MPLS при використанні комутації по міткам (Label Switching Шлях або скорочено LSP) являє собою результат, що призводить до розподілу мітки між деяким LSR.

Таблиця, що встановлює співвідношення в парах «вхідний інтерфейс та вхідна мітка» та «префікс адреси одержувача, вихідний інтерфейс та вихідна мітка» мають всі маршрутизатори. Будь-який Label Switching маршрутизатор, що отримав пакети, призначає використавши номери інтерфейсів, на котрі

вже прийшли макети в яких є вже прив’язані мітки, вихідний інтерфейс. Тому нові значення міток, які базуються в розділі «вихідна мітка» таблиці, змінюються на старі значення, і пакети далі будуть передаватися до наступних чергових пристроїв шляхом LSR [9]. З цього виходить, що для одноразових ідентифікацій значень полів в одному рядку таблиці та необхідна для цієї операції. На все це потрібно мало використаного часу чим при звичайній, традиційній маршрутизації, де порівнюють IP-адреси відправників з найбільшими довгими адресними префіксами в таблиці маршрутизації.

Ядра та граничні області — це дві області, за підтримкою яких розподіляють мережі MPLS та мають різні функції. Головні вимоги до пристроїв, які створюють ядро - це підтримка MPLS та процеси маршрутизацій потоку інформації для тих протоколів, які виконують комутації за MPLS. Комутацію можуть брати на себе виключно всі маршрутизатори ядра. Полем діяльності граничних LSR вважаються всі функції класифікації пакеті з різними FEC, реалізація додаткові сервіси такі, як фільтрація, маршрутизація, вирівнювання навантаження та управління трафіком. З цього виходить, що в граничній області виконуються інтенсивні обчислення, а на ядро припадає високопродуктивна комутація, яка відкриває шлях до оптимізування конфігурації пристроїв MPLS. В результаті, головною особливістю цієї технології є роз’єднання аналізу IP-адреса в заголовку пакета від процесу його комутації, що призводить до безлічі цікавих можливостей.

Технологія MPLS являє собою незалежний від яких завгодно протоколів і масштабований механізм передачі даних. Мітки присвоюються пакетам даних у мережах, що ґрунтуються на ній. Дозвіл про пересилку пакета даних до іншого вузлу MPLS виконується, за умовою, якщо мітці буде присвоєно значення. Отже це відкриває можливість створити незалежний від середовища передачі наскрізний віртуальний канал і застосовувати будь-який протокол.

Управління трафіком (TE), підтримка класів обслуговування (CoS) та віртуальних приватних мереж (VPN) — це три головні галузи використання

MPLS в мережах значних постачальників мультисервісних послуг на сьогоднішний час.

1. Управління трафіком;

Traffic Engineering — це одне з найважливіших застосувань MPLS. Головним показником високоякісної діяльності мультисервісної мережі стає гарантія додатків, або виконанню всіх вимог до комплекту характеристик якості сервісу трафіку користувачів. До комплекту характеристик відмінного обслуговування трафіку відносять: максимальний час доставки, розкид часу доставки і ймовірність втрат. Взагалі виникає потреба в механізмах підтримки QoS.

Одним із способів виконання такої задачі являється оптимальне користування наявних ресурсів мережі. Головним завданням підвищеної ефективності є якісне регулювання пропускної спроможності. Все це призводить до зменшення перевантажень.

Головними чинниками, які створюють перевантаження є:

- недостатність ресурсів або їх невідповідність існуючої навантаженні;
- неефективність розподілу потоків трафіку по наявним ресурсам.

Якщо не вистачає ресурсів мережі для подолання перевантаження необхідно або збільшення ресурсу, або використання традиційних засобів управління перевантаженням, або об’єднання цих двох способів.

До зниження рівня потреби до існуючого в мережі рівня ресурсів приводять традиційні класичні способи управління перевантаженням.

Як механізми застосовуються обмеження потоків, управління чергами в маршрутизаторі, управління шириною вікна для потоку, диспетчеризація і ін..

Проблеми, зумовлені неефективністю розподілу трафіку по ресурсам мережі, можуть бути вирішені шляхом управління трафіком із застосуванням політики балансування навантаження в різних фрагментах мережі.

Балансування визначає політику оптимізації робочих характеристик мережі з метою мінімізації перевантаження. Коли перевантаження мінімізована шляхом оптимального використання ресурсів, затримки і втрати пакетів зменшуються, а сукупна пропускна здатність зростає. Це призводить до того, що якість мережевого обслуговування для кінцевого користувача поліпшується.

2. Підтримка класів обслуговування;

Користувачі мережі загального користування мають потребу в мультисервісності мереж для різноманітних програм і цілей таких як, наприклад, чутливість до затримок передачі голосу і відео, або невибагливої електронної пошти. Для задоволення вимог, сервіс-провайдер користується, як способами управління трафіком, так і коштами для його класифікації.

Таким чином технологія спроможна допомогти провайдерам передавати пакети з урахуванням класу обслуговування (CoS) .

В мережі MPLS існує два способи. В першому способі надається можливість обробки пакетів у вихідних чергах LSR-маршрутизаторів з урахуванням основних значень, які містяться в заголовку MPLS. У другому способі, кожна пара, яка є сукупністю вхідного і вихідного LSR-маршрутизаторів, має декілька LSP-маршрутів з різними ознаками таких як продуктивність, смуга пропускання, час затримки та інших. Після цього вхідний маршрутизатор переправляє один тип трафіку по одному LSP-маршруту і т. д.

3. Віртуальні приватні мережі;

Суттю VPN являється те, що вона створює модель роботи мережі, яка є корпоративною та територіально розподіленою. Підтримкою якої є пакетна мережа доступна для загального використання. Можливість індивідуального і колективного доступу до корпоративної мережі практично в будь-який час швидко перетворюється на уряд вимога ділового світу. Це і породило нову тенденцію серед підприємств застосовувати пакетні мережі для організації глобальних корпоративних зв’язків з метою об’єднання окремих локальних мереж філій.

Отже явну загрозу становить передача даних публічною пакетною мережею, такою як Інтернет. Тому обов'язково потрібно налагоджувати безпечну передачу даних для будь-яких організацій.

Внутрішні ресурси корпоративної мережі стають доступними для користувачів мережі загального користування, а конфіденційний трафік може бути переглянутий зловмисниками. Взагалі і рівень продуктивності, надійності та якості обслуговування притаманні каналам публічного зв'язку бажають кращого.

Ці проблеми вирішуються за допомогою VPN. За основу беруть ідею використання мереж загального вжитку для захищеної пересилки трафіку територіально віддалених філій замовника, а застосуванням ідеології побудови приватних мереж. В даний час цю послугу широко потребують підприємства і організації, які не мають власних мережевих ресурсів.

Основною причиною появи і використання VPN була і залишається їх відносно низька вартість: компаніям дешевше скористатися послугами провайдерів по створенню і незалежного супроводу розподілених корпоративних мереж, ніж витрачати гроші на побудову власних глобальних мереж.

Технологія MPLS здатна використовувати та реалізовувати послуги VPN нового покоління. Її здатність до якісної пересилки по IP-мережам, передачі трафіку, який є чутливим до затримок і є основою для створення в корпоративних мережах нових послуг телекомунікацій - зокрема, як передача в реальному часі голосу і відеозображення. А користування такими сервісами, як відеотелефонія та відеоконференції стає можливим завдяки MPLS VPN.

Для MPLS VPN притаманно взаємопроникнення з такими сервісами, як Інтернет, веб, поштові служби і хостинг, а також великає масштабованість і використання автоматичної конфігурації. Завдяки цьому MPLS VPN ексклюзивно різниться від інших форм побудови віртуальних приватних мереж. Кількість провідних провайдерів послуг, що пропонують своїм клієнтам скористатися сервісом MPLS VPN для економічного побудови мереж Internet і Ex-

tranet, стає дедалі більше, роблячи даний сервіс доступним для користувачів дедалі більшої кількості країн і регіонів.

Крім клієнтів VPN загальна мережа обслуговує достатньо велику кількість інших користувачів. Отже, захист віртуальної мережі клієнта є одним з пріоритетних завдань до обслуговування VPN. Захист мережі включає такі пункти, як, по-перше, не допустимість розкрадання трафіку користувачів при відправленні через мережу провайдера, по-друге, не дати клієнту зі злими умислами доступу до ресурсів, які розміщені на сайтах інших користувачів, по-третє, неуможливити розкриття топології користувача.

Технологія многопротокольної комутації міток для VPN надає ту ж ступінь захисту, що і з'єднання другого рівня, але без складнощів, властивих шифрування. Оскільки при пересиланні даних по мережі MPLS IP-адреси не використовуються, внутрішні адреси можна зробити конфіденційними.

В сучасних IP VPN організовуються захищені прямі тунелі. В результаті, у міру збільшення числа вузлів віртуальної мережі, виникає проблема масштабування. До того ж, корпоративним користувачам доводиться управляти функціонуванням цих мереж. У мережах MPLS VPN функціонування віртуальної мережі забезпечує провайдер послуг, а значить корпоративним клієнтам немає потреби в їх управлінні. При збільшенні нових вузлів VPN не виникає проблем з масштабуванням, завдяки тому, що вони створюються на основі архітектури MPLS.

1.2 MPLS мітки

Міткою називають короткий елемент, який має фіксовану довжину, і застосовується для локального ідентифікації класу еквівалентності пересилання FEC. На рисунку 1.1 зображено структуру мітки. Вона має такі параметри: 4 байта (32 біта) – довжина, 20 біт – заголовок, 12 біт – підзаголовок.

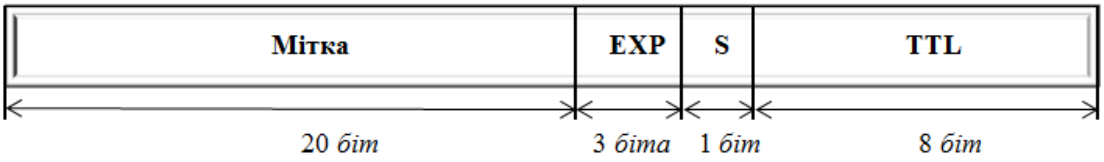

Рисунок 1.1 – Структура MPLS мітки [2]

Заголовок мітки складається з наступного:

1. мітка використовується для вибору відповідного маршруту комутації по мітках;

2. поле експериментальних бітів (EXP), які тримають у резерві майбутні експеременти і дослідження;

3. поле MPLS-стека (S) вважається способом підтримки ієрархічності структури стека міток. Остання мітка в заголовку має параметр 1, а в у всіх інших залишившихся мітках в стеку дорівнює 0;

4. час життя (TTL) дублює аналогічне поле IP-пакета, яке є засобом скидання пакетів в мережі внаслідок утворення закільцьованих маршрутів.

З мітками проводять три типи операцій:

1. label push (додавання мітки) як правило проводиться маршрутизатором за допомогою вхідного LER. Мітка, яка буде визначена, і вихідний інтерфейс, через який буде передаватися пакет, визначаються на базі таблиці IP-маршрутизації для вхідного IP-пакета;

2. label swap (комутація по мітці) здійснюється за підтримкою маршрутизатора LSR. Проходячи обробку на основі таблиці міток, вхідні пакети з мітками, здійснюють передачу пакета зі зміною мітки через поставлений інтерфейс. При цьому ймовірне приєднання інших міток до стеку;

3. label pop (зняття мітки) здійснюється вихідним LER маршрутизатором або тим, який був попереднім йому LSR маршрутизатором. На базі таблиці маршрутизації, якби мітка була останньою в стеку, то пакет данних пересилається вже крайньому одержувачу без міток.

Пакет, що має вигляд стеку дозволяє мати більше однієї мітки, що дає альтернативу створювати відповідність міток. На рисунку 1.2 показано, як виглядає дворівневий стек MPLS-міток.

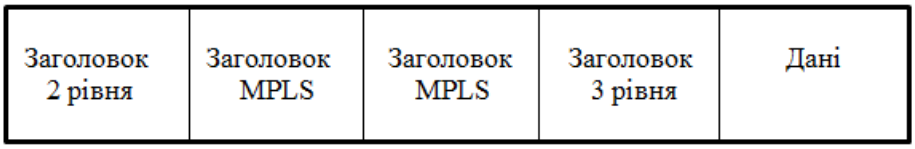

Рисунок 1.2 – Дворівневий стек міток MPLS

Можливість об’єднання кількох LSP стає можливим завдяки функціям стека MPLS. Створюється агрегований тракт MPLS в результаті додавання зверху мітки до стеку кожного шляху в технології. В місці, де закінчується цей тракт, він роз’єднується на частини його індивідуальних тунелей. Також тракти, що мають спільну складову маршруту, можуть об’єднуватися. Це означає, що MPLS має здатність забезпечувати ієрархічну відправку даних. А це є найважливішою і найзатребуваною функціональною можливістю. Якщо її застосувати, то глобальну маршрутну інформацію не має сенсу переносити. В результаті мережа MPLS стає найбільш стабільною в порівнянні з іншими мережами, які мають традиційну маршрутизацію.

Число міток, які використовуються дорівнює 1048576. Резервовані мітки мають значення від 0 до 16.

- 0 – «IPv4 Explicit null Label». MPLS-мітка, яка знаходиться на дні стека. Робота її полягає в тому, що вона інформує, що треба витягнути пакета стек, а також шлях цього пакета повинен базуватися на заголовку IPv4 [11];
- 1 «Router Alert Label». Ця мітка окрім дна стека, може знаходитись будь-де в стеці міток. Якщо в пакеті, що прибуває, мітка розташовується вгорі стека, то його перенаправляють місцевому програмному модулю для обробку. Надалі маршрут буде проведений чи мітці, що знаходиться нижче, чи на основі інформації, яка закладена в заголовок протоколу мережевого рівня або інкапсульованих в нього протоколів. Мітка «Router Alert Label» при переміщенні пакета знову розміщується нагорі стека [11];

- 3 «Implicit null Label». Задача LSR полягає в привласненні і поширенні цієї мітки, але вони не будуть вказані пакетами. При необхідності заміни мітки, яка розташовується вгорі стека, причому новою міткою є «Implicit NULL Label», то LSR робить видалення стека міток, замість того, щоб провести операцію заміни [11];
- 4 — 15 — значення резерву для майбутнього виконання [11].

1.3 Протокол LDP

Протокол розподілу міток (Label Distribution Protocol) – протокол, за підтримкою якого два маршрутизатора комутовані по міткам в мережі MPLS ділиться інформацією про відображення міток. Обмін інформацією здійснюється за двома напрямами. LDP створений для побудови цілісних маршрутів комутації по мітках LSP.

Робота протоколу розпочинається зі створення тунелю комутованого по міткам тракту, який створює співвідношення до наявних записів в таблицях маршрутизації про шляхи в мережі. Цей протокол вибирає лише певні замітки таблиці маршрутизації, якими є утворення на базі протоколів, які розташовані всередині маршрутизації, або протоколів типу IGP. MPLS IGP режим – порядок формування механічного шляху MPLS за участі LDP.

Він призначається у відповідність FEC будь-якому LSP, що був їм створений. Клас еквівалентної пересилки поєднаний з тунелем MPLS вказує, який пакет наступний з цим LSP. Шлях MPLS проходить через мережу так, що будь-який комутуючий мітки маршрутизатор відповідає за з’єднання вхідної мітки для FEC з вихідною міткою, що визначає наступний кроку для певного FEC.

Треба конкретно знати і розуміти, які пакети відповідають кожному LSP. Відбувається це завдяки специфікації FEC для будь-якого шляху MPLS. Клас еквівалентної переадресації розпізнає всі потрібні пакети, що можуть бути поєднанні з LSP.

Описати кожен FEC можна, як сукупність з одного або більше елементів. Будь-який FEC-елемент вибирає набір пакетів, які є можливість поєднати з певним LSP. Тунель, що використовує кілька елементів FEC, то такий LSP має кінець у вузлі або ще раніше, причому елементи FEC йдуть один за одним в одному напрямі.

Існують наступні види елементів класу еквівалентної переадресації:

- префікс адреси. Даний елемент має довільну довжину від 0 до повної адреси включно;
- адреса ЕОМ *(*електронно-обчислювальна машина*)*.Представлений елемент – повна адреса ЕОМ.

Результат призначення міток базується на певних мірах пересилання, ними являються одноадресна маршрутизація до одержувача, оптимізація розподілу трафіку в мережі, багатоадресна розсилка, VPN, механізми забезпечення QoS. Програмне забезпечення специфікації протоколу LDP встановлює принципи, за якими визначаються відповідність між вхідним пакетом і його LSP.

Порядок, за яким створюється зв’язок пакета з комутованим по міткам трактом містить такі правила:

1. є один LSP, в якому закладено FEC елемент адреси електронно-обчислювальної машини, розпізнаний адресою місця призначення пакета, значить пакет об’єднується з даними LSP;

2. є декілька LSP, в яких FEC закладено елемент адреси ЕОМ, який розпізнається адресою призначення пакета, значить пакет об’єднується з даними LSP.

3. пакет повністю відповідає одному LSP, пакет повністю об’єднується з цим шляхом;

4. пакет відповідає деяким LSP, він об’єднується з трактом, у знаючи що у LSP довший префікс. Коли довшого префікса не має можливості ідентифікувати, то пакет з’єднується з LSP, в якого префікс найдовший з усіх.

5. якщо напевно знати, що пакет проходить через визначений маршрутизатор, тобто LSP, який містить клас еквівалентної пересилки адресного префікса, це також є адресою даного маршрутизатора, тоді пакет об’єднується з цим LSP.

Ці всі правила виконуються за процедурою, доки пакет буде відправлено до даного тунелю.

Побудовою MPLS визначається значення мітки, що значить її прив’язку до вибраного FEC, розробляє комутуючий по міткам маршрутизатор, котрий виявляється приграничним шляхопровідником для пакетів даних цього класу еквівалентності. Ці LSR мають назви downstream LSR (нижчий) і upstream LSR (вищий). Отже, downstream повідомляє сусідні верхні LSR про те, що присвоїв мітки до кожного FEC пересилаються до нього пакети. Така процедура має назву – розподіл міток.

Порядок протоколу LDP дає дозвіл LSR, які реалізовують даний протокол, здійснювати LSP. Останнім об’єктом цього тракту є або сусідній LSR, який зв’язаний з таким LSR, або вихідний LSR, до якого даний комутуючий по міткам маршрутизатор зможе дістатися через декілька транзитних LSR.

Як правило розподіл міток обслуговує downstream LSR, але ініціювати цей процес не завжди може він. Запустити розподіл міток може LSR, пересилаючи до downstream LSR доцільний запит. Назва цього режиму – downstream «on-demand». Будь-який розподіл міток виникає в двох випадках, або після запитів зверху, або з ініціювання downstream LSR. Назва цього режиму – «unsolicited downstream».

Downstream LSR здійснює розподіл міток не завжди по upstream LSR, котрі зв’язані з ним понапрямку. LDP застосовують і для встановлення спілкування між двома LSR, попри те, що вони об’єднані лише мережею. Але кінцевий ітог розподілу в даному випадку напряму залежить в якому статусі, знаходиться LSR, в консервативному або ліберальному.

Порядок консервативного розподілу міток полягає в тому, що до повідомлення прив'язується мітка, яке приймають не від поруч знаходжений LSR, це не береться до розгляду та відкидається. Маршрутизатор з підтримкою комутації по міткам з'єднує мітку до FEC лише коли, він являється вихідним маршрутизатором або коли він одержав повідомлення про прив'язку від сусіднього з ним LSR. Завдяки цьому режиму обробляється меншу кількість міток.

Сутністю ліберального режиму є прив'язка мітки, випускається нижчим LSR, з ним відсутнє і з'єднання напряму, береться до уваги та фіксується і користується. Цей режим задовольняє тим, що при зміні форми розташування мережі співвідношення між міткою та FEC не змінюється, навіть коли зв'язок з LSR, що встановлює це взаємозв'язок, набуває не комутованого, а прямого характеру. Але ліберальний режим має і мінуси в роботі. Це насамперед те, що на обробку та зберігання на вищий LSR припадає багато відомостей про відношення між мітками і FEC.

LDP має чотири типи повідомлень:

1. discovery messages (повідомлення виявлення), які застосовуються при оголошенні підтримки наявності в мережі маршрутизатора з підтримкою комутації по міткам;

2. session messages (сеансові повідомлення), які використовую для ть відкриттів, допомоги і зупинки LDP-процедур між LSR;

3. advertisement messages (повідомлення оголошення), які застосовують для відкриттів, заміни і розірвання прив'язаної мітки до FEC;

4. notification messages (оповіщення), що несуть додаткові повідомлення, а також повідомлення про хибні дії та недоліки.

Для ефективної роботи протоколу зобов'язується перевірка і налагодженість повідомлень. Для забезпечення норм LDP застосовують для якісної передачі даних TCP протокол для повідомлень сесій, застережень і рекламування. Отже, будь-яке свопування даних створюється на UDP (протокол датаграм користувача).

Структура LDP-повідомлень ґрунтується на наступних принципах:

1. повідомлення LDP відправляються в блоки протокольних даних (PDU) по TCP-з'єднанню протягом LDP сесії. Всякий PDU пересилає декілька повідомлень, при цьому допускається, що повідомлення в окремому PDU можуть не об'єднуватися один з одним. На рисунку 1.3 показано як візуально виглядає LDP PDU;

Рисунок 1.3 – Візуальний вигляд LDP PDU

2. LDP PDU має заголовок протоколу розподілу міток, за ним йдуть LDP-повідомлення. Визначено три поля, з яких складається заголовок: версія, довжина і ідентифікатор. Віртуальний показ заголовка продемонстровано на рисунку 1.4;

Рисунок 1.4 – Візуальний вигляд заголовка LDP

Докладніше ознайомлення з полями заголовка є наступним:

- «версія» – поле, яке має довжину 2 байти і ціле число, номер модифікації протоколу;
- «довжина PDU» – поле, яке займає довжину 2 байти і відповідає лише за сукупну відстань PDU в байтах, відкидаючи поля версії і відстані;

- «ідентифікатор LDP» – поле, яке заповнює 6 байт і однозначно розпізнає простір міток маршрутизатора з підтримкою комутації відправника, який PDU не застосовує.

Перші 4 байти розпізнають LSR і зобов'язані бути масштабно одним-єдиним, неповторними. Повністю визначений id маршрутизатор повинен займати 32 біта, призначений LSR, який застосовується для розпізнавання при демодуляції вузлів у вектора тракту. Два байти, які залишилися, дають змогу розпізнати простір міток вказаного LSR. Нульове значення мають 2 байти, які скеровані на платформу для простору міток.

3. Структура LDP повідомлення має 6 полів:

- «біт U» – це біт невідомого повідомлення. При одержані незнайомого повідомлення, якщо біт U = 0, як відповідь відправнику відправляється повідомлення, якщо біт U = 1, незнайоме повідомлення нехтується;
- «вид повідомлення» – це поле, що розпізнає вид повідомлення;
- «довжина повідомлення» – це поле, яке відповідає за сукупний розмір в байтах полів розпізнавача повідомлення;
- «id повідомлення» – код, який займає 32 біти і застосовується для розпізнання повідомлень. Відправник застосовує LSR для мінімізування та розпізнавання повідомлень. Маршрутизатор з підтримкою комутації по міткам, відправляє повідомлення у відгук на дане повідомлення, який обов'язково впускає цей id в TLV статус, і передається даним повідомленням.
- «обов'язкові параметри» – це сукупність обов'язкових показників, в яких змінюється розмір. Є повідомлення, які не містять необхідних показників взагалі;
- «опціонні параметри» – це сукупність повідомлень, які можуть мати всі необхідні параметри, що змінюють свій розмір. Для повідомлень притаманні опціонні параметри, ці показники не йдуть послідовно один за одним.

На рисунку 1.5 показано загальну структуру LDP повідомлення.

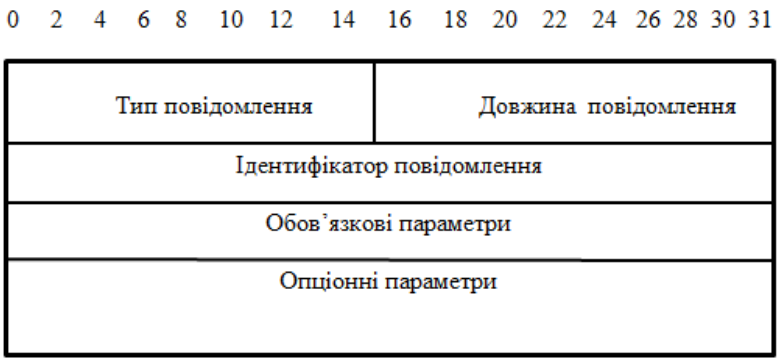

Рисунок 1.5 – Формат LDP повідомлення

Протокол LDP містить пристосування для уникання зацикленості. Застосовують це для того, щоб не допустити зациклення шляхів і запитів. Цей результат можна отримати, якщо в повідомлення Label Mapping і Label Mapping Request вставляти інформацію про LSR через конкретні запити, що пройшли перевірку. Якщо LSR-маршрутизатори працюють в статусі Ordered Control уникнення зациклення отримати не складе сил. Коли ж LSR користуються Independed Control, це означає в даному обсязі LSR необхідно знову пересилати повідомлення і відповіді на них, бо ж всі дані про LSR-маршрутизаторах, через які відсилались запити, будуть оновлені. Механізм уникнення зацикленості не обов'язково вживати, так як за задумом гарантом запобігання зацикленності виступає протокол IP-маршрутизації, даними якого користується протокол LDP. Якщо такі проблеми не мають місця, то не надовго і якщо процедура протоколу маршрутизації нешвидко сходить, LDP працює швидше, ніж протокол IP-маршрутизації.

1.4 Протокол OSPF

OSPF — найпопулярніший протокол на базі сукупності ознак і положення каналу. Цей протокол маршрутизації, який не містить розподілу на класи. Користуючись алгоритмом Дейкстри, за допомогою процесу засобів і методів збору, накопичення, обробки і передачі даних відслідковують характер каналу і відшукують короткі маршрути. Протокол OSPF містить найскладнішу конфігурацію основи даних. Протоколи на основі вектора довжини не мають даних про мережі, які знаходяться на значній відстані, то прото-

кол на базі стану каналу підтримує усі дані про маршрутизатори і їх об'єднання, які вимкнуті. Положення каналу в цьому протоколі характеризує інтерфейс (IP-адрес, маска, тип мережі і т.п.) та його взаємодію з маршрутизаторами, які є сусідніми.

На базі характерних ознак інтерфейсів складається основа інформацію про положення каналів (Link-State Advertisement – LSA), які відправляються регулярно чи після змінення конфігурації мережі, чи яких-завгодно змінеснях на маршрутизаторах. Цими повідомленнями є маленькі пакети. LSA мають дані про підключених інтерфейсах, метриках і інших вимірах. На базі прийнятих LSA повідомлень маршрутизатор застосовує послідовність SPF, який створює дерево коротких шляхів. Алгоритм розробляє обчислення над інформацією про конфігурацію мережі, викидаючи непотрібні гілки (шляхи). Створені шляхи вносять в таблицю маршрутизації.

Протокол OSPF вважається внутрішнім протоколом маршрутизації і функціонує в окремій автономній системі. Її можна розбити на частини, які являють собою наслідовні області автономної системи.

На рисунку 1.6 розглядається дворівнева мережна наслідуваність, в якій маршрутизатори функціонують по своїм правилам. Демонстрація однієї автономної системи, у якої є лише один граничний маршрутизатор (№0), який забезпечує взаємодію з іншими мережами.

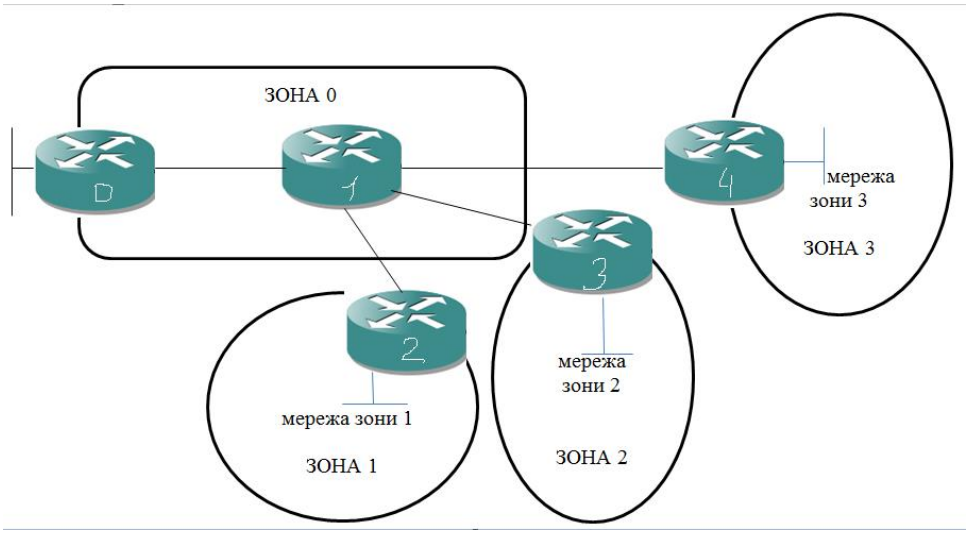

Рисунок 1.6 – Дворівнева мережна ієрархія

Автономна зона має в своєму складі:

1. нульова область – зона, що відповідає за з'єднання всіх областей, її назва – магістральна область. В цій зоні маршрутизатор №1 має назву магістральним;

2. області 1-3 – маршрутизатори в них не є магістральними. Тож маршрутизаторам цих областей відомо конфігурацію зони, де вони розташовані, і займаються базою даних станів каналів;

3. грань 0 та 1-3 зон – це зони, в яких розташовуються граничні маршрутизатори №2-№4. Вони слідкують за базою даних положень каналів усіх зон, к яким підключені;

4. маршрутизатор №0 є граничним, тільки для автономної системи.

В повноваження протоколу входять такі функції:

- створює граничні стосунки з сусідами;
- відправляє LSA для створення на будь-якому маршрутизаторі бази даних;
- приводить в дію SPF для обчислення найкращих шляхів до кожного отримувача;
- наповнює таблицю маршрутизації найкращими шляхами до кожного отримувача.

OSPF створює граничні відносини з сусідами таким чином. Двум маршрутизаторам, які мають спільну границю і виконують протокол OSPF, необхідно вислідити один-одного в мережі, раніше ніж починається обмін даними між ними. Цей процес виконується за допомогою протоколу Hello. Маршрутизатори, в яких встановлений OSPF, з кожних інтерфейсів відправляють Hello-пакети.

Протокол OSPF має такі переваги:

1. циклічними не можуть бути маршрути, обраховані цим протоколом;

2. протокол гарантує масштабованість для великих мереж;

3. найвисоке переобладнання при зміні топології мережі.

Недоліками OSPF є:

1. топологія має вигляд ієрархії;

2. недостатність розподілу навантаження при неповноцінних маршрутах;

3. використовування метрики лише для оцінки шляху.

Для бізнесу використання даного протоколу дає наступні переваги:

- відмовостійкість. У випадки виходу з ладу будь-якого з маршрутизаторів, обмін інформацією миттєво перемикається на інший маршрут, що запобігає простою в роботі;
- економія. Зв'язок між вузлами надійно резервується, а зміна структури не вимагає великих трудових втрат. Таким чином не потрібно утримувати багато персоналу, який буде обслуговувати систему;
- зниження ризиків. Використання даної технології значно знижує ризики простою, а також ризик залежності функціонування системи від обслуговуючого персоналу.

1.5 Постановка задачі

В даний час Інтернет спрощує життя людини, стаючи необхідністю. Із такою популярністю Всесвітньої мережі почався розвиток і популярність різноманітних веб-додатків, які теж полегшують життя сучасній людині.

Зробивши аналіз літературного огляду можна цілком приступати до створення власної розробки веб- орієнтованої системи. Графічний інтерфейс, якої дозволить налаштовувати інтерфейси маршрутизаторів у автоматичному режимі та динамічну маршрутизацію за протоколами OSPF, LDP. Вимогою до системи є те, що завдяки ній легко та просто буде можливість переносити згенерований код налаштувань до симулятора GNS3.

Розробка графічного інтерфейсу повинна починатися зі створення вебсторінки, на якій можна буде вводити вхідні дані та отримувати результат налаштувань. Великою перевагою буде те, що результати налаштувань мож-

на буде одразу ж копіювати і вносити в GNS3. У подальшому буде одержано налаштовані маршрутизатори, а потім й повноцінну мережу.

Наведено ряд задач, які треба виконати:

1. розробка конфігурації мережі на основі графічного симулятора GNS3;
2. створення графічного інтерфейсу, який налаштує протоколи OSPF, LDP;
3. тестування отриманих конфігурацій з розробленої веб-орієнтованої системи в симуляторі GNS.

2 МОДЕЛЮВАННЯ ТЕХНОЛОГІЇ MPLS З ВИКОРИСТАННЯМ СИМУЛЯТОРА GNS3

2.1 Графічний симулятор GNS3

На сьогоднішній день існує безліч симуляторів для побудови та налаштування комп'ютерних мереж, такі як UNetLab, Cisco Packet Tracer, але перевагою GNS3 перед цими симуляторами є те, що в GNS3 підтримуються всі функції як на реальному обладнанні.

GNS3 (Graphical Network Simulator) — це програмне забезпечення, яке дозволяє змоделювати роботу пристроїв мережі передачі даних. Він є незамінним інструментом для того, щоб навчатися. Роботу може здійснювати на всіх платформах. Відмінно підходить для створення тестів на desktop-машинах.

Найцікавішим фактором GNS3 є можливість з'єднання проектованої мережі з реальною мережею. Тож це є унікальною можливістю перевірки на практиці будь-якого проекта, без залучення реального обладнання. GNS3 не потребує оплати. Він являється відкритим програмним забезпеченням, і будь-який бажаючий може завантажити його. Кросплатформність – це одна з особливостей симулятора. Тому фішкою GNS3 є використання його на більшості платформах та підтримка популярних ОС.

До недоліків даного симулятора відносять: відсутність повноцінної симуляції комутаторів 2-рівня Cisco, високі вимоги до системних ресурсів і дефекти.

Програму WireShark застосовують для проведення оцінки трафіку в спроектованій мережі.

Середовище GNS3 використовує наступне:

1. WinPCAP — системний драйвер і бібліотека функцій, що дозволяє отримати доступ до мережевих інтерфейсів фізичного комп'ютера і

переданої / одержуваної інформації по ним. Використовується для аналізу трафіку, що передається по мережі;

2. Wireshark — графічний аналізатор мережевого трафіку. Дозволяє наочно відобразити докладну інформацію про мережевий трафік. Використовується як всередині середовища GNS3, так і дозволяє аналізувати трафік з реальної комп'ютерної мережі;

3. Dynamips — середовище моделювання мережевих пристроїв, реалізованих на основі процесорів з MIPS архітектурою. Щоб функціонувати вимагає наявності образів ОС iOS мережевих пристроїв CISCO. Допускається виконання і інших операційних систем;

4. VCPS, VirtualBox, QEMU — середовища моделювання ЕОМ. Використовуються для емулювання кінцевих мережевих пристроїв або проміжних пристроїв, реалізованих на базі ЕОМ з архітектурою IBM / PC;

5. SuperPUTTY — система віртуальних терміналів. Дозволяє підключатися до мережевих пристроїв для управління ними.

2.2 Розробка схеми в GNS3

Робота над проектом починається з головного вікна симулятора GNS3, який показаний на рисунку 2.1.

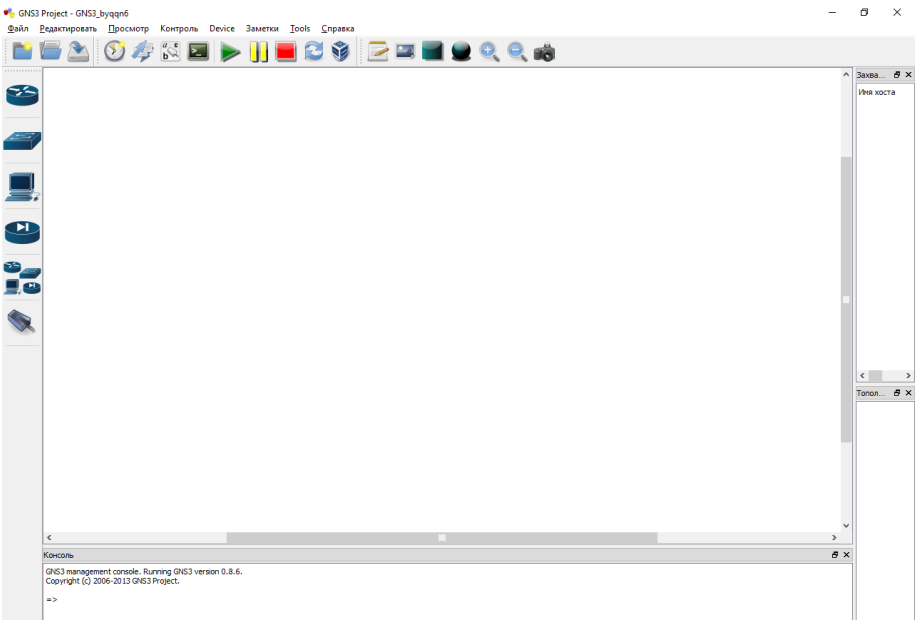

Рисунок 2.1 – Головне вікно середовища GNS3

За допомогою панелі пристроїв додаємо їх на робочий простір, з’єднуючи між собою (рис. 2.2).

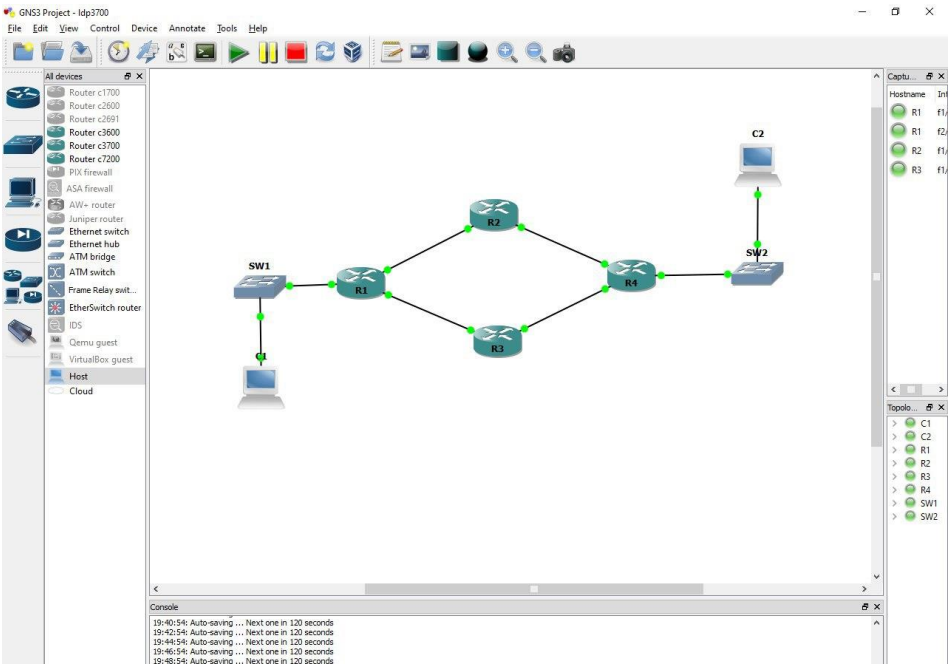

Рисунок 2.2 – Прототип мережі MPLS

По-перше, мережу MPLS було створено в середовище GNS3. По-друге, здійснено налаштування маршрутизаторів (вказане IP на кожному з портів, який використовується), а також налаштування протоколів OSPF та LDP. При розробці мережі використовувались маршрутизатори 3700, комутатори та віртуальні комп’ютери.

На рисунку 2.3 продемонстровано конфігурацію протоколу OSPF на маршрутизаторі R1 в GNS3:

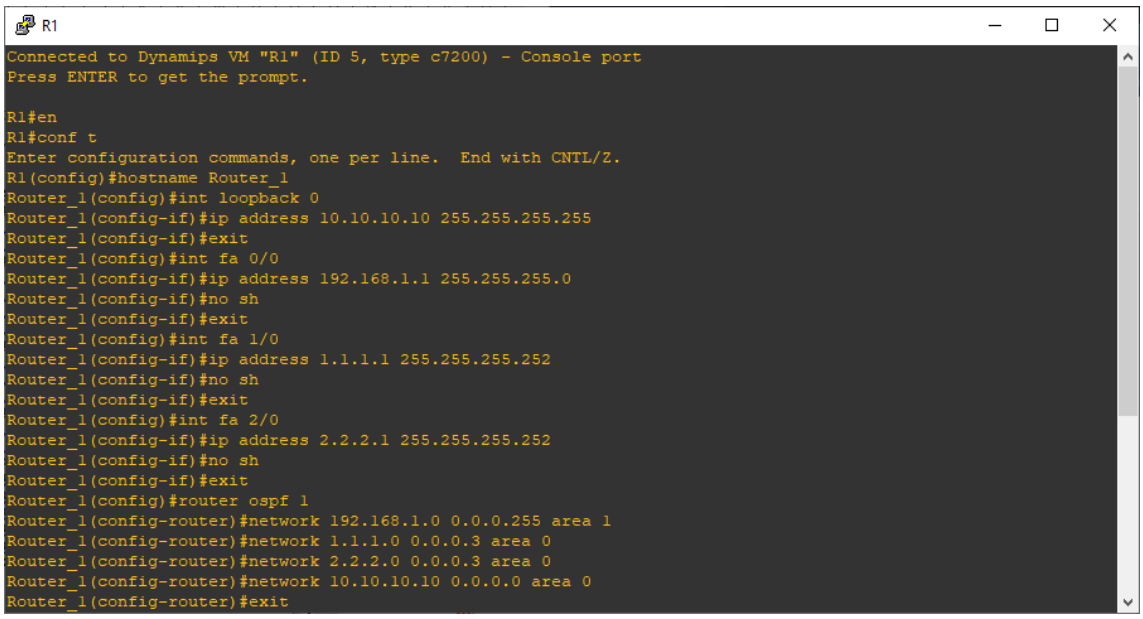

Рисунок 2.3 – Конфігурація протоколу OSPF в симуляторі GNS3

Конфігурація протоколу LDP на маршрутизаторі R1 в GNS3 визначається наступними командами (рис. 2.4).

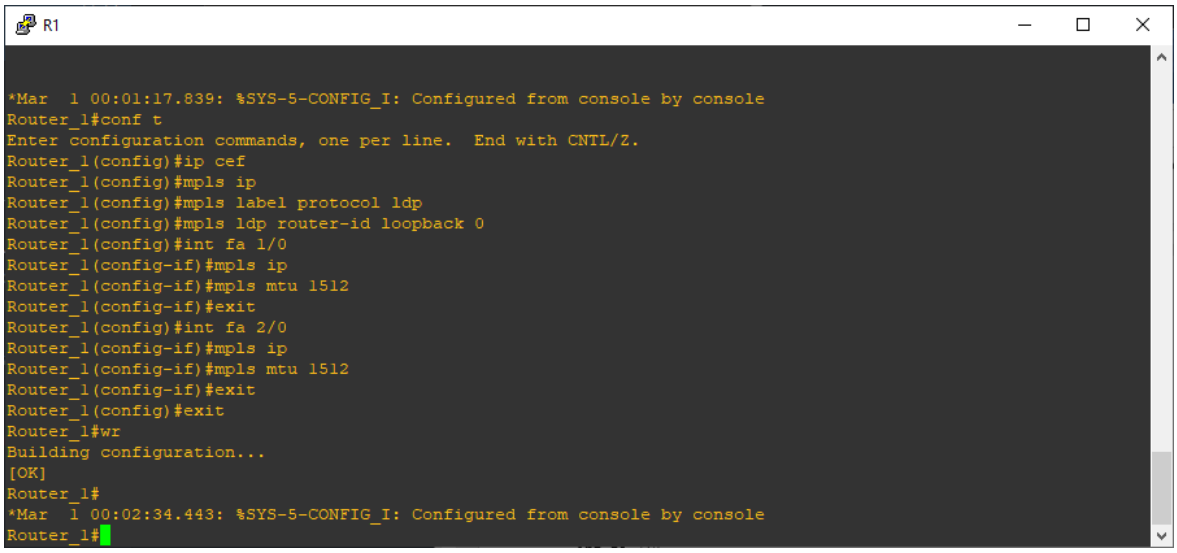

Рисунок 2.4 – Конфігурація протоколу LDP в симуляторі GNS3

Маршрутизатори R2, R3, R4 були налаштовані відповідно.

Перевірка роботи здійснюється за допомогою команди ping. На рисунку 2.5 продемонстровано з’єднання маршрутизатора R1 з R2. Команда ping — це інструмент для перевірки цілісності з’єднань на основі TCP / IP.

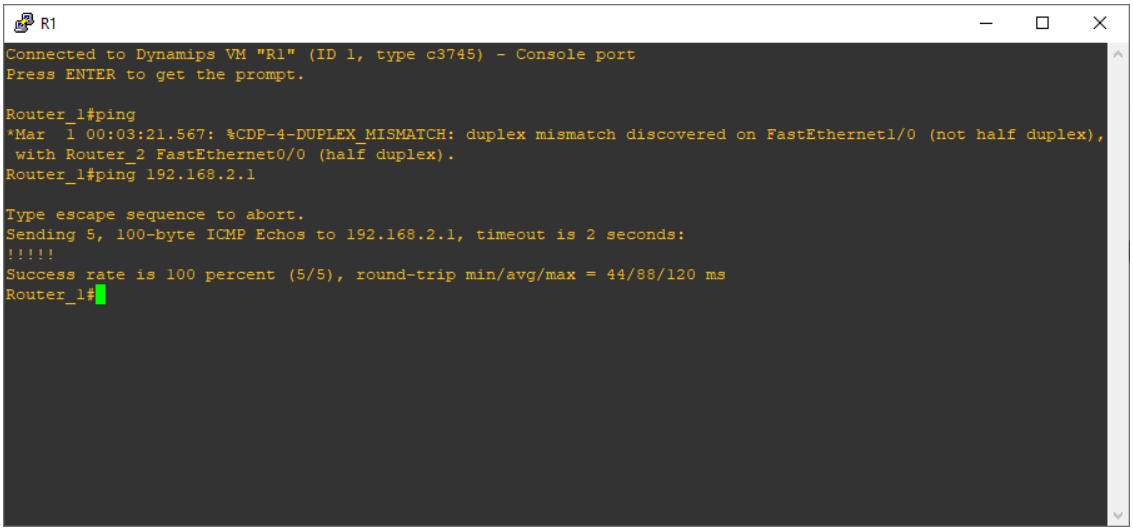

Рисунок 2.5 – Робота команди ping в GNS3

Прототип готової схеми показано на рисунку 2.6.

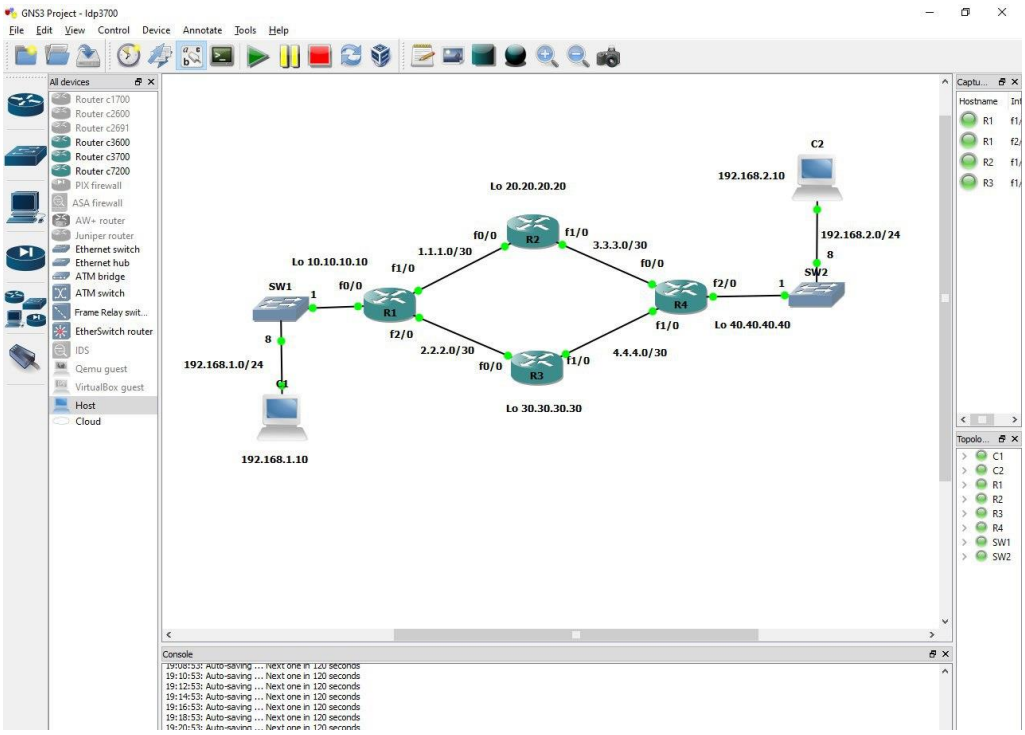

Рисунок 2.6 – Прототип готової схеми

Для захоплення та аналіза мережевого трафіка застосовуємо програму Wireshark. Усі пакети мають власне кольорове маркування. Наприклад, пакети протокола ICMP, позначені рожевим кольором. (Рисунок 2.7).

ICMP — це один з протоколів мережевого рівня в моделі ISO / OSI. Його задачею є обслуговування функції контролю правильності роботи мережі.

ICMP пакети з’являються тоді, коли відбувається зв’язок від одного комп’ютера до іншого.

В закладці Multiprotocol Label Switching Header показані різні параметри такі,як значення мітки (Label), час життя (TTL) і т.і).

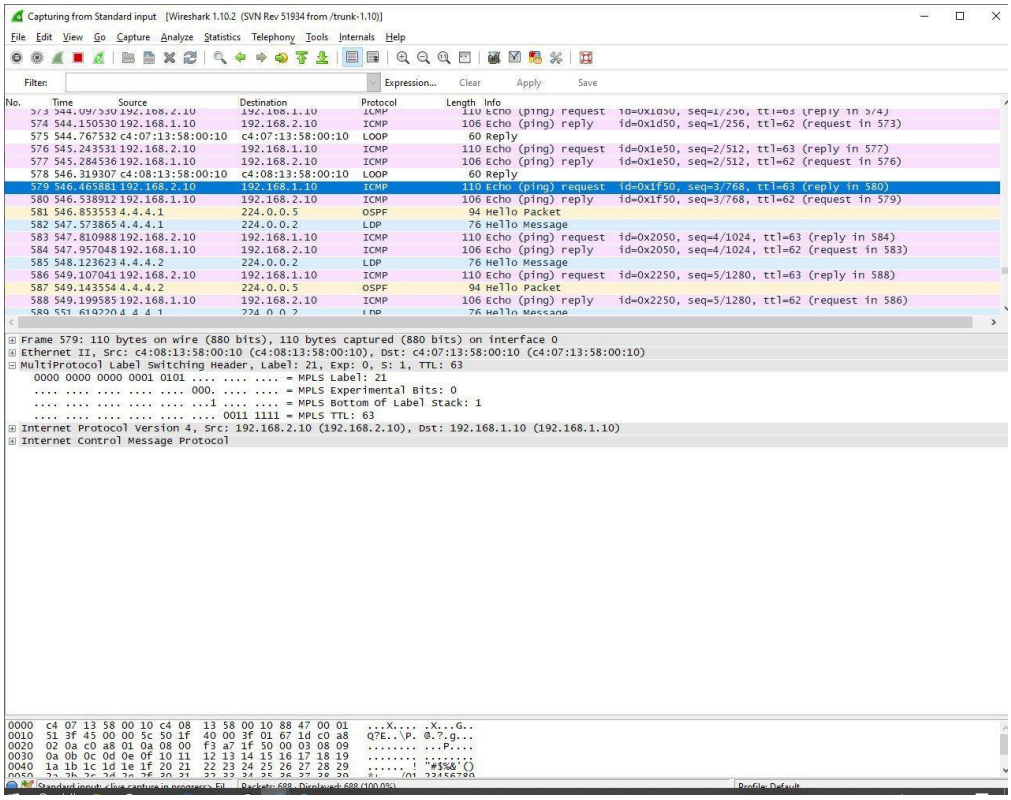

Рисунок 2.7 – Головне вікно Wireshark

Для наочності в програмі Wireshark можна зробити фільтрацію захоплених пакетів. Наприклад, відфільтрувати лише пакети протоколу LDP, як показано на рисунку 2.8. Ці пакети відсилаються маршрутизаторам на multicast адресу 224.0.0.2.

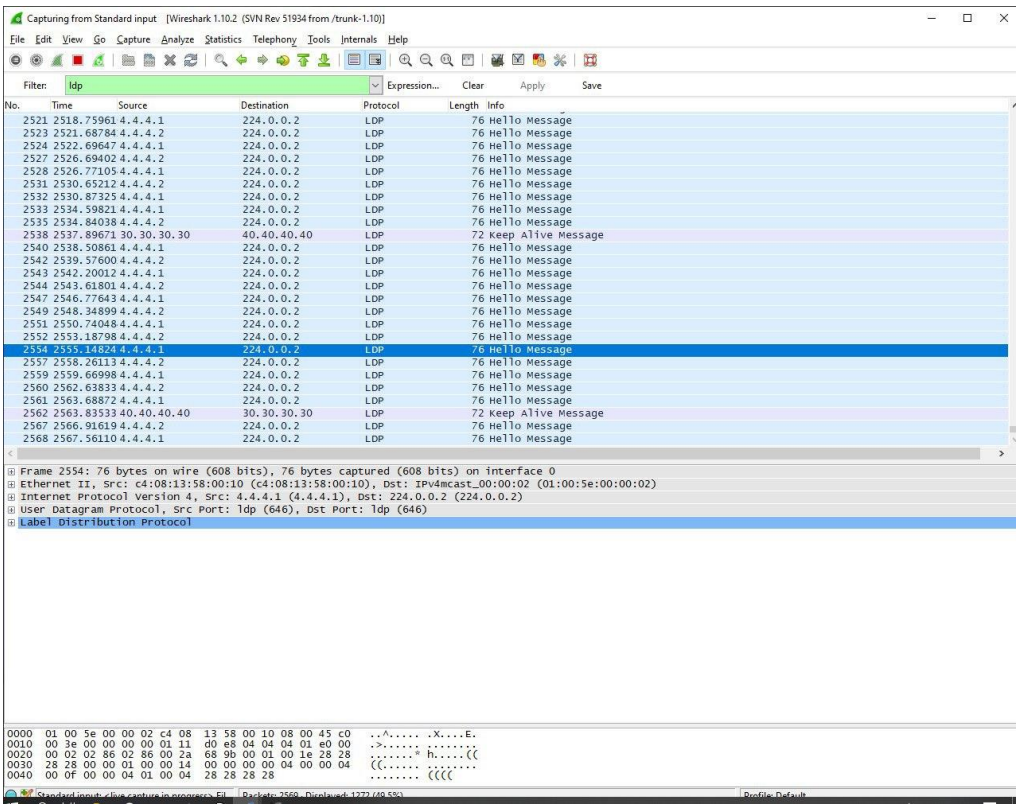

Рисунок 2.8 – Фільтрація пакетів протоколу LDP в Wireshark

3 РОЗРОБКА ГРАФІЧНОГО ІНТЕРФЕЙСУ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ

3.1 Засоби розробки веб-інтерфейсу

Саме при реалізації дипломного завдання були задіяно засоби HTML, CSS та JavaScript. Головною задачею мов є миттєва і ефективна організація інтерфейсу користувача, доступу до інформації, безпечне користування мережею.

Для розробки сучасного та функціонального інтерфейсу використовують наступні мови веб-програмування:

1. HTML (Hypertext Markup Language) є мовою розмітки гіпертексту в основі розташовується World Wide Web (Всесвітньої Павутини). Мова HTML має можливість розташувати будь-який текст, трансформувавши його в гіпертекст з наступним виданням у мережі. Показ сторінки веб-браузером забезпечується наборами символів мови HTML. Символи мають назву дескриптори, до їх складу входять елементи для створення гіперпосилань.

Особливою відмінністю HTML-документів полягає в тому, що документ має містити лише текст, а за підтримки особливих тегів об’єкти вміщуються в документ у період його показу браузером і утримуються нарізно.

2. CSS (Cascading Style Sheets) – це процес налагодження вебсторінок, задля розподілу змісту сторінки і її зовнішнього вигляду. Цей розподіл доступний: можна, не змінюючи вмісту сторінки, переоформити стиль її вигляду. Каскадні таблиці уживають для надання кольору, шрифтів, розміщення елементів на сторінці. Внутрішньою таблицею стилів вважають, таблицю стилів, яку можна вбудувати прямо в HTML-сторінку. А відповідно зовнішньою називають, таблицю стилів, яку її можна розробляють в окремому файлі, і згодом додають посилання на нього до потрібної HTML-сторінки.

3. Мова програмування JavaScript, створена фірмою Netscape для розробки інтерактивних HTML-документів. Вона є об’єктно-орієнтованою

мовою створення вмонтованих додатків, що застосовується клієнтом, так і сервері. Набір правил, що описуються, подібний на синтаксис Java.

Області JavaScript розподіляють на наступне:

- динамічна розробка документа за підтримки сценарію;
- своєчасний контроль правдоподібності внесених даних користувачем до пересилки їх на сервер;
- розробка динамічних html-сторінок сумісно з CSS та об'єктною моделлю документа;
- взаємозв'язок при розв'язанні периферійних задач користувачем, що виконуються за допомогою додатка JavaScript, який вмонтовується в html-сторінку.

3.2 Розробка графічного інтерфейсу та налаштування протоколів OSPF, LDP

В графічному симуляторі GNS3 була зконфігурована мережа MPLS, налаштовані маршрутизатори і динамічна маршрутизація за протоколами OSPF та LDP.

Браузер слугує основою для роботи з системою. Запустивши файл Diplom.html з'являється наглядний вигляд графічного інтерфейсу (рис. 3.1). Отже після відкриття сторінки, користувач може побачити мережу MPLS, протоколи динамічної маршрутизації: OSPF,LDP та розташування кнопок «Згенерувати» та «Скопіювати». Вигляд кнопок – radio buttons. Інтерфейс наочно зрозумілий для всіх. Тож кожен зможе користуватися ним.

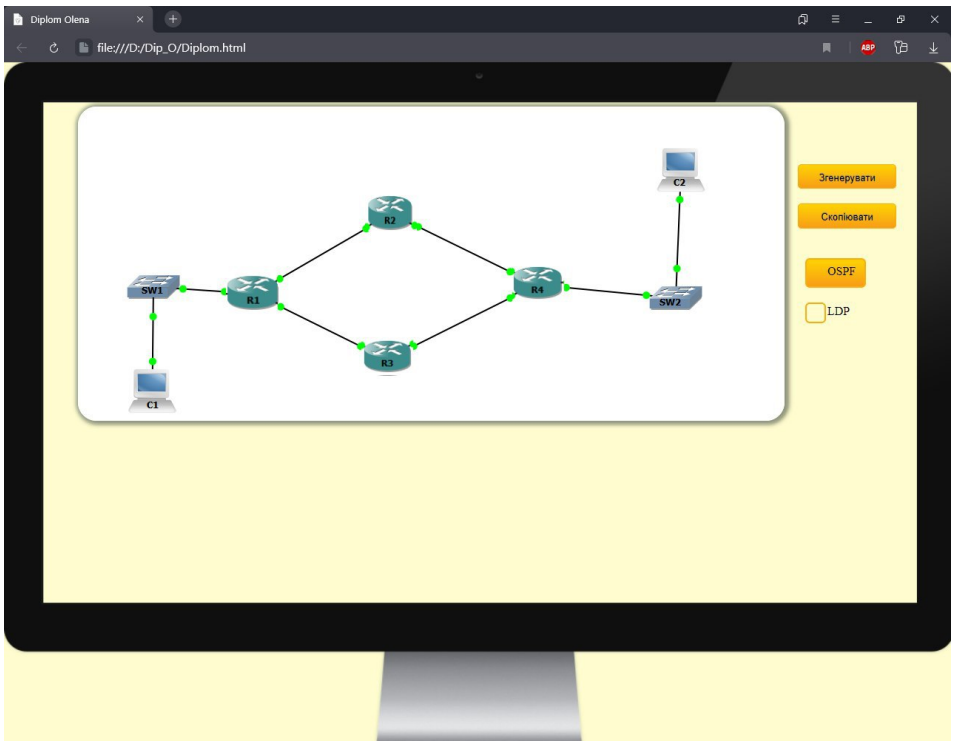

Рисунок 3.1 – Веб-інтерфейс системи

Робота розпочинається з того, що при натисканні на обраний маршрутизатор, з’являются поля для вводу IP,MASK та LOOP (рис. 3.2, 3.3).

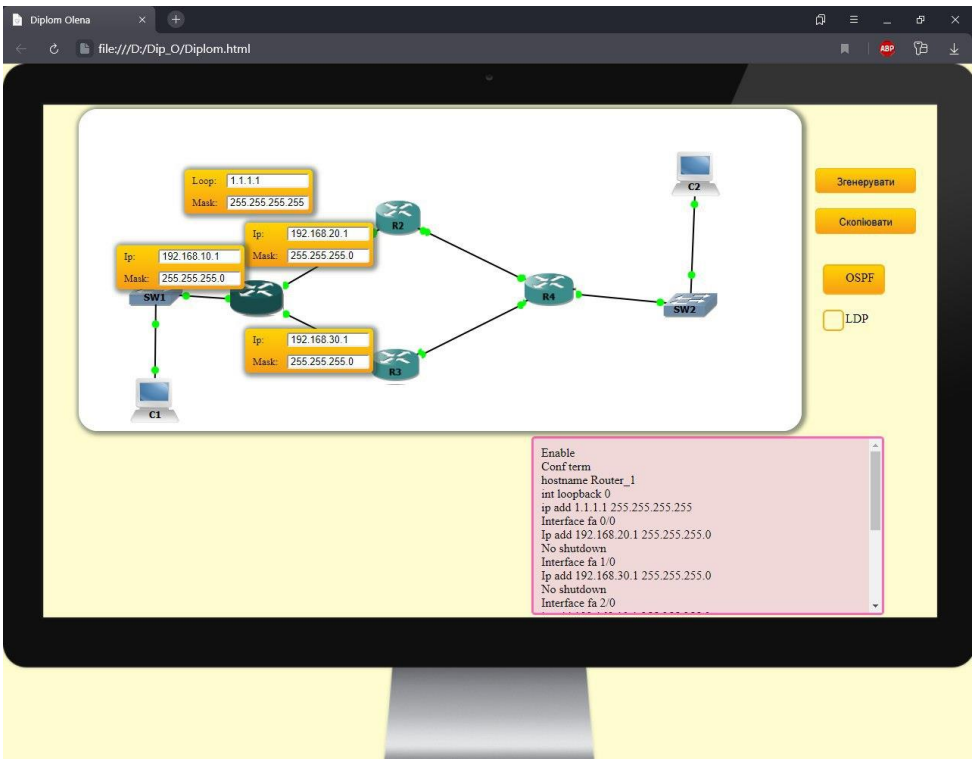

Рисунок 3.2 – Налаштування маршрутизатора за протоколом OSPF у вікні браузера

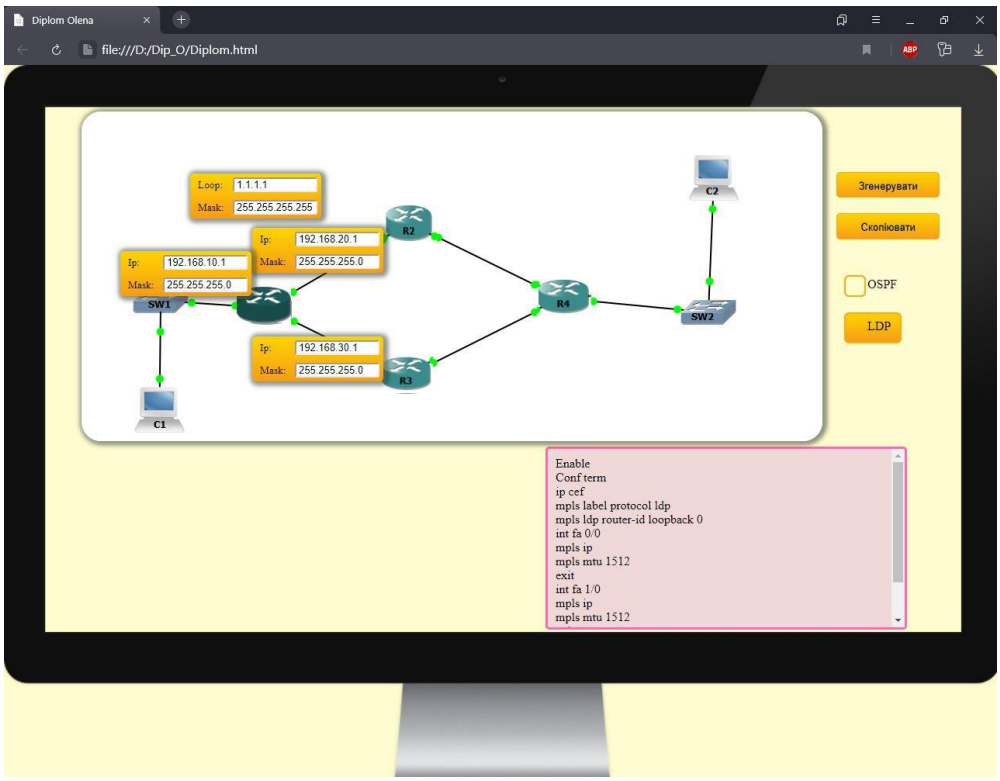

Рисунок 3.3 – Налаштування маршрутизатора за протоколом LDP у вікні браузера

Призначення кнопки «Скопіювати» таке, вона копіює вміст випадаючого вікна програми для подальшого використання в симуляторі GNS3.

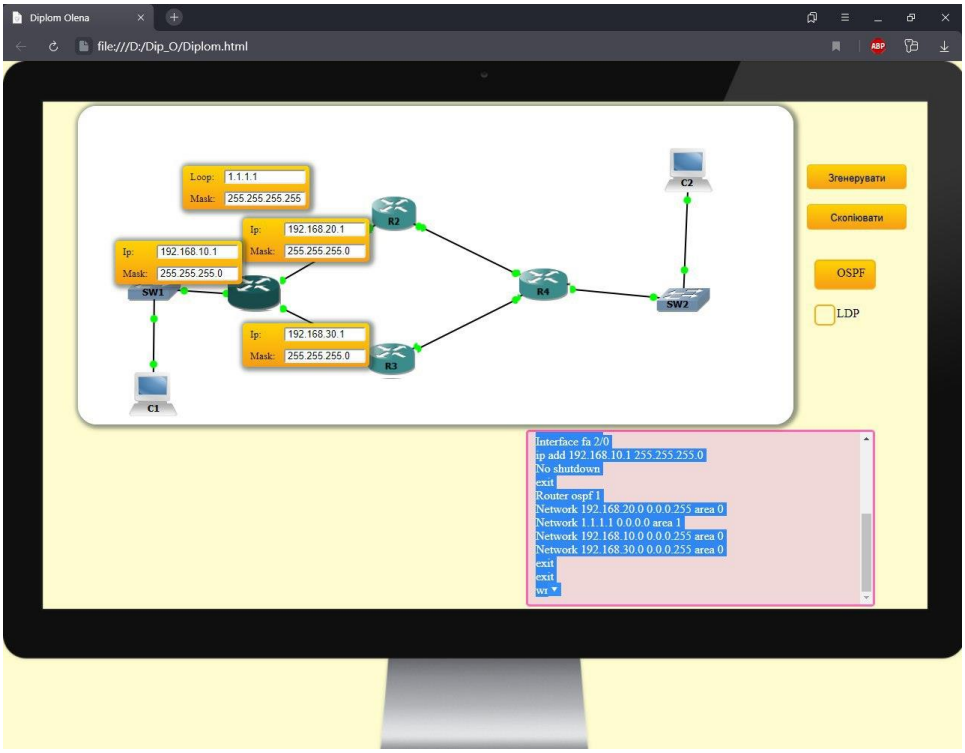

Рисунок 3.4 – Робота кнопки «Скопіювати»

Усвідомившись, що все введено правильно і всі поля заповнені, користувач обирає за яким протоколом зробити конфігурацію. В роботі дані на вибір два протоколи динамічної маршрутизації: OSPF і LDP. Обравши потрібний, треба натиснути кнопку «Згенерувати». Вона потрібна, щоб згенерувати код. Після натискання цієї кнопки з'являється вікно резутьтата з прокруткою.

Головною перевагою є те, що при подальшій конфігурації дані налаштувань попереднього маршрутизатора не втрачаються.

3.3 Перевірка працездатності в GNS3

Для перевірки роботи треба відкрити сторінку в браузері і заповнити відповідні поля обраного маршрутизатора, вибравши при цьому протокол та натиснути «Згенерувати». Після цього скопіювати код з вікна результата за допомогою натиснення кнопки «Скопіювати». Зазделегідь треба запустити симулятор GNS3 зі створеною схемою, а потім після запуску відкрити консоль маршрутизатора, який треба налаштувати і вставити скопійовані код. Інші маршрутизатори на схемі були зконфігуровані від самого початку.

Налаштування будуть проводитись на маршрутизаторі, який має назву R3(рис. 3.5-3.8).

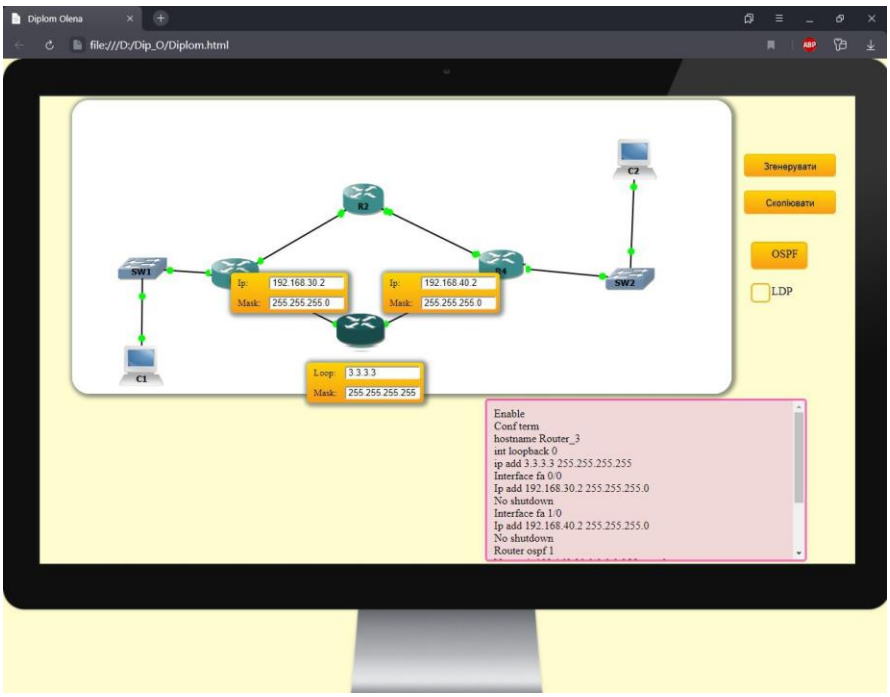

Рисунок 3.5 – Налаштування за протоколом OSPF у вікні браузера

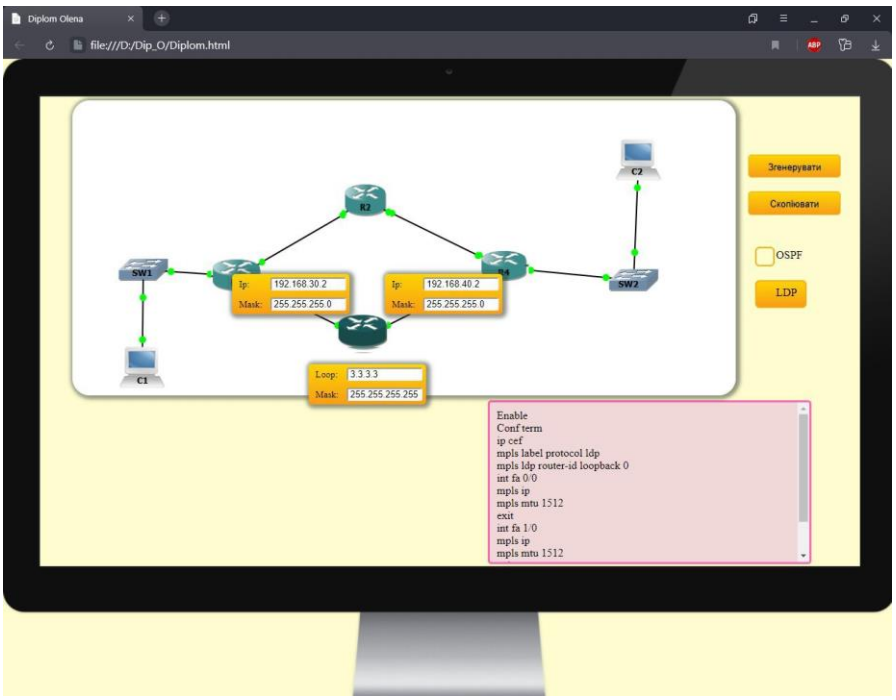

Рисунок 3.6 – Налаштування за протоколом LDP у вікні браузера

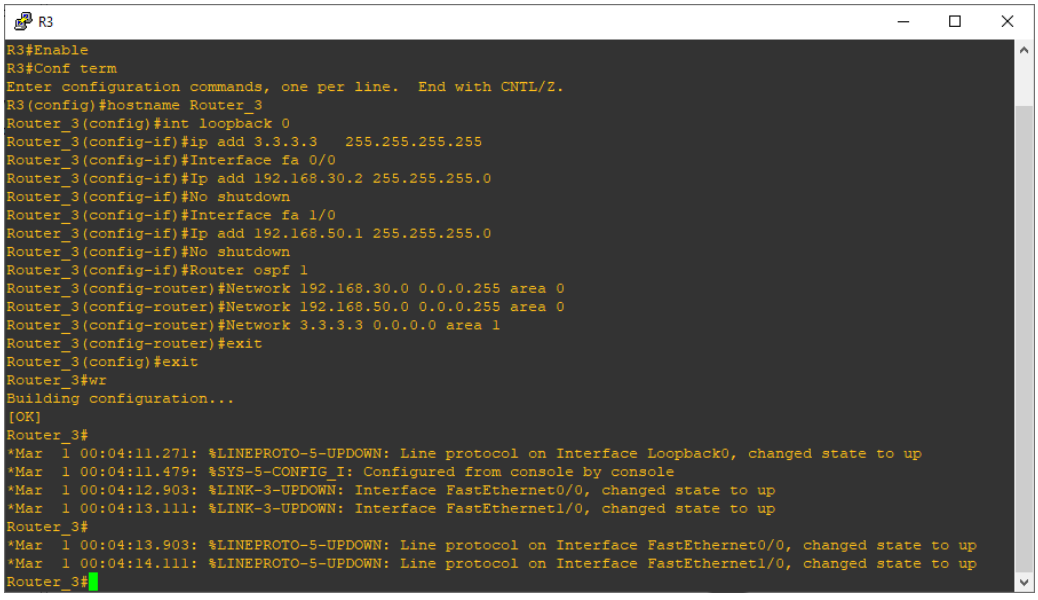

Рисунок 3.7 – Конфігурація Router_3 за протоколом OSPF в GNS3

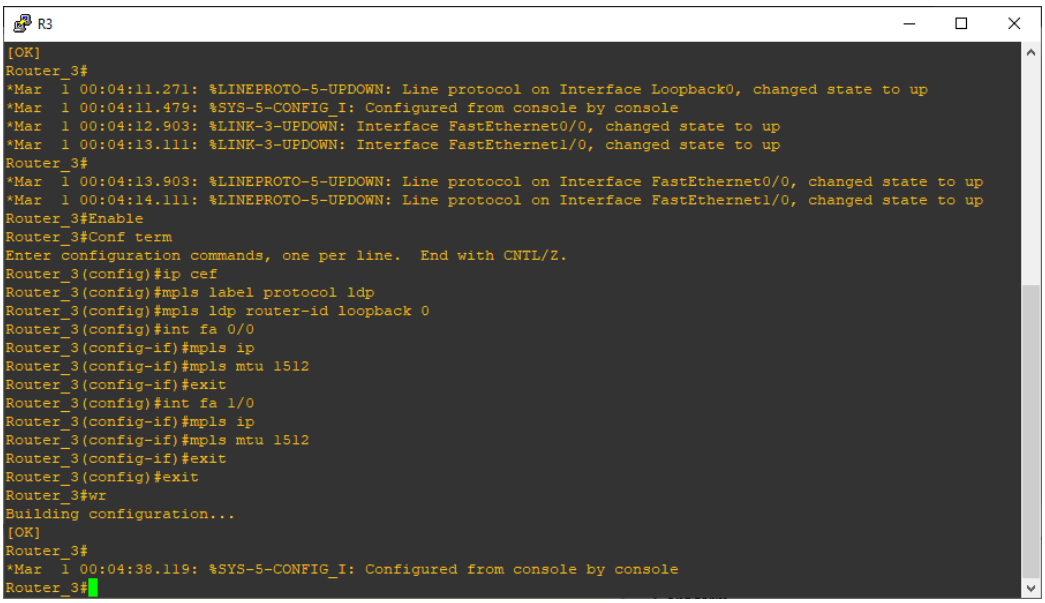

Рисунок 3.8 – Конфігурація Router_3 за протоколом LDP в GNS3

При перевірці працездатності схеми в GNS3 шляхом пінгування Router_3 з Router_2, результат є вдалим, а отже все працює справно (рис. 3.9).

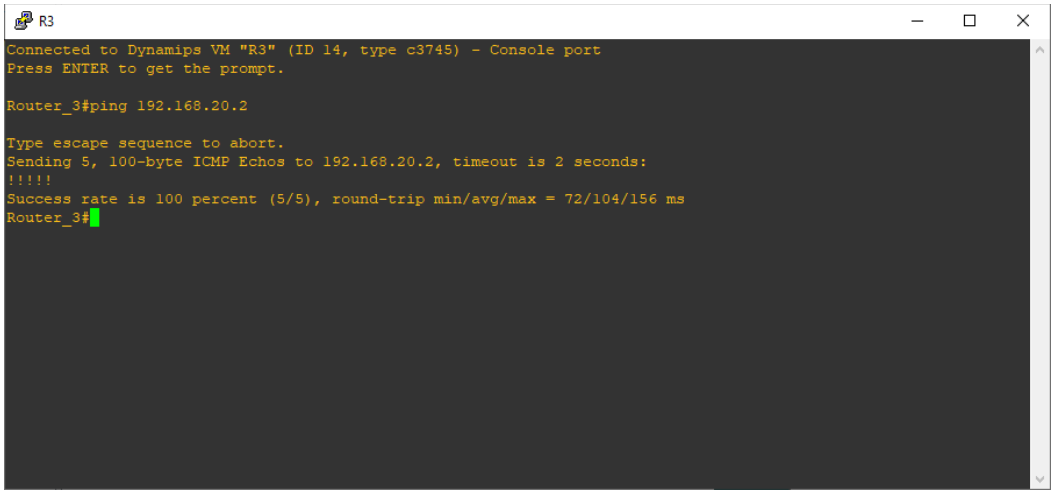

Рисунок 3.8 – Робота команди ping в GNS3

Після тестування веб-інтерфейсу стало зрозуміло, що він працює як і планувалося, тому створений інтерфейс в майбутньому може бути застосований як для налаштування внутрішніх протоколів LDP та OSPF в симуляторах, так невдовзі та й на справжньому обладнанні.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра з'ясовано принцип роботи одного з найпопулярніших мережевих симуляторів — GNS3. Найважливішим фактором GNS3 є можливість з'єднання проектованої мережі з реальною мережею. GNS3 дозволяє дослідити властивості створеної MPLS мережі без залучення реального обладнання. Кросплатформність – це одна з особливостей симулятора. Тому фішкою GNS3 є використання його на більшості платформах та підтримка популярних ОС. Модуль WireShark співвпрацює з GNS3 аби здійснити аналіз трафіка на відповідність поставленій задачі. Побудована мережа демонструє адекватну топологію та можливість швидкої реалізації її на практиці.

Технологія MPLS вважається однією з провідних транспортних технологій. Принцип роботи MPLS складається з того, що протоколи маршрутизації застосовують для з'ясування топологічних мереж. Проте задля переміщення даних в границі мереж визначається один з постачальників послуг , який використується технікою віртуальних каналів. В дипломній роботі проведено аналіз та налаштування внутрішніх протоколів маршрутизації таких, як LDP та OSPF. Призначення протоколу LDP полягає у побудові цілісних маршрутів комутації по мітках LSP. А протокол OSPF використовується для поширення даних маршрутизації всередині однієї автономної системи.

Результатом роботи є створена веб-орієнтована інформаційна система. В інтерфейсі якої дозволяється налаштування на маршрутизаторах CISCO конфігурації та динамічної маршрутизації за одним з обраних протоколів (LDP, OSPF). Починаючи роботу з програмою для маршрутизаторів задаються ключові параметри такі, як IP-адреси інтерфейсів і маски підмереж, а потім вибирається протокол динамічної маршрутизації. Згенерований код налаштування динамічної маршрутизації можливо використовувати на реальному обладнанні в майбутньому.

СПИСОК ЛІТЕРАТУРИ

1. Олифер В.Г., Олифер Н.А Компьютерные сети. Принципы, технологии, протоколы (5-е издание) – СПб.: Питер, 2016. — 992с.

2. Эффективный механизм передачи данных в опорных IP-сетях с использованием технологии MPLS [Электронний ресурс] - https://wireless-e.ru/gsm/mpls/

3. 10.4.6. Сигнальні протоколи резервування мережних ресурсів [Електронний ресурс] - http://www.znanius.com/3819.html

4. Сети MPLS [Электронний ресурс] - https://ru.bmstu.wiki/Сети_MPLS

5. протокол LDP [Электронний ресурс] - http://um.co.ua/1/1-8/1-82962.html

6. Принцип роботи мережі MPLS [Електронний ресурс] - http://ni.biz.ua/7/7_9/7_90395_printsip-raboti-seti-MPLS.html

7. Програми для створення сайтів. Редактори таблиць CSS [Електронний ресурс] - https://webstudio2u.net/ua/programming/189-css-redactor.html

8. Руководство по проектированию OSPF [Электронний ресурс] - https://www.cisco.com/c/ru_ru/support/docs/ip/open-shortest-path-first-ospf/7039-1.html

9. СТРУКТУРА МЕРЕЖІ НА БАЗІ ТЕХНОЛОГІЇ WiMAX [Електронний ресурс] - https://infopedia.su/16x12476.html

10. Технологія MPLS [Електронний ресурс] - https://wiki.cuspu.edu.ua/index.php/Технологія_MPLS

11. Сети для самых маленьких. Часть десятая. Базовый MPLS [Электронний ресурс] - https://habr.com/en/post/246425/

ДОДАТКИ

Додаток А

```
<html>
<head><title>Diplom Olena</title>

<script
src="https://cdn.rawgit.com/zenorocha/clipboard.js/master/dist/clipboa
rd.min.js"></script>

<script src="js/Dip.js" type="text/javascript"></script>
<link rel="stylesheet" href="css/styleDip.css">

<style>
body {
    background:  url(pictures/1.png);
    background-size: 1285px 1000px;
    background-color: #fffdd0;
   }
 </style>
</head>
          <img class="topology"   src="pictures/fon.png">
     <div class="routers">
          <div class="noblow">
               <img src="pictures/Rout1.png" class="marsh Rout1">
               <img src="pictures/Rout2.png" class="marsh Rout2">
               <img src="pictures/Rout3.png" class="marsh Rout3">
               <img src="pictures/Rout4.png" class="marsh Rout4">

          </div>
          <div class="blow">
               <img src="pictures/r0.0.png" class="marsh Rout1">
               <img src="pictures/r0.0.png" class="marsh Rout2">
               <img src="pictures/r0.0.png" class="marsh Rout3">
               <img src="pictures/r0.0.png" class="marsh Rout4">

          </div>
     </div>



     <input value="Згенерувати" class="generete button"
type="button"><br>
     <button value="Скопіювати" class="copy-btn button" data-
clipboard-target="#code">Скопіювати</button>
     <div class="radio-buttons">
          <div>
          <input type="radio" name="merega-type" id="radio1" val-
ue="ospf" checked/>
               <label for="radio1">OSPF</label>

          </div><br>
          <div>
               <input type="radio" name="merega-type" id="radio2"
value="ldp">
```

```
                <label for="radio2">LDP</label>
            </div>

      </div>
            <div class="block Rout1">
                  <div class="add-m add-m-1">
                        <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                        <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>
                  <div class="add-m add-m-2">
                        <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                        <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>
                  <div class="add-m add-m-3">
                        <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                        <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>
                  <div class="add-m add-m-4">
                        <div><label class="add-lab">loop:
</label><input  class="adds" ></div>
                        <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>

            </div>


            <div class="block Rout2">


                  <div class="add-m add-m-1">
                              <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                              <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>
                  <div class="add-m add-m-2">
                              <div><label class="add-lab">ip:
</label><input  class="adds" ></div>

                              <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>
                  <div class="add-m add-m-3">
                        <div><label class="add-lab">loop:
</label><input  class="adds" ></div>
                        <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                  </div>

            </div>
```

```
          <div class="block Rout3">


                    <div class="add-m add-m-1">
                         <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>
                    <div class="add-m add-m-2">
                         <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>
                    <div class="add-m add-m-3">
                         <div><label class="add-lab">loop:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>

          </div>


          <div class="block Rout4">


                    <div class="add-m add-m-1">
                         <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>
                    <div class="add-m add-m-2">
                         <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>
                    <div class="add-m add-m-3">
                         <div><label class="add-lab">ip:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>
                    <div class="add-m add-m-4">
                         <div><label class="add-lab">loop:
</label><input  class="adds" ></div>
                         <div><label class="add-lab">mask:
</label><input  class="maska" ></div>
                    </div>

          </div>


     <div class="gen-code-b"">
```

```
            <div class="code" id="code">

            </div>

      </div>


<script>

new ClipboardJS('.copy-btn');


$('.blow .marsh').click(function(e){
      var classes = $(e.currentTarget)[0].classList
      var tabClass = '.'+classes[1]

      hideAllBlocks();
      hideAllRouters();
});

$('.noblow .marsh').click(function(e){
      $('.right-content').show();

      var classes = $(e.currentTarget)[0].classList
      var tabClass = '.'+classes[1]

      hideAllBlocks();
      hideAllRouters();

      showBlock(tabClass);
      showRouter(tabClass);
});

function showBlock(block){

$(block).css('display', 'block')
$(block).addClass('active')
}

function showRouter(router){
      $('.blow '+router).css('display', 'block')

      showCodeBlockIfNeeded();
}

function hideAllRouters(){
      $('.blow .marsh').css('display', 'none')
}

function hideAllBlocks(){
$('.block').css('display', 'none')
$('.block').removeClass('active')
}

$('.generete').click(showCodeBlockIfNeeded)

function showCodeBlockIfNeeded(){
      var isValid = inputData();
```

```
      if (isValid) {
            showCodeBlock();
      } else {
            hideCodeBlock();
      }
}

function inputData(){
      var ipMasksBlocks = $('.block.active .add-m');
      var wroteIPs = []

      ipMasksBlocks.each(function(i, element){
            var ipMasksBlock = ipMasksBlocks[i];

            var ip = $(element).find('.adds').val()
            var mask = $(element).find('.maska').val()

            wroteIPs.push({'adds': ip, 'maska':mask});
      })
      return wroteIPs
}

 function showCodeBlock()  {
      $('.gen-code-b').show();
      generateCode();
 }

 function hideCodeBlock()  {
      $('.gen-code-b').hide();
 }


// __ip_N_ __network_N_ __mask_N_ __back_maska_N_ __ip_net_N_
var codes = {
      "Rout1": {
            'ospf': [
                  "Enable",
                  "Conf term",
                  "hostname Router_1",
                  "int loopback 0",
                  "ip add  __adds_4_   __mask_4_",
                  "Interface fa 0/0",
                  "Ip add __adds_1_ __mask_1_",
                  "No shutdown",
                  "Interface fa 1/0",
                  "Ip add __adds_2_ __mask_2_",
                  "No shutdown",
                  "Interface fa 2/0",
                  "ip add __adds_3_ __mask_3_",
                  "No shutdown",
                  "exit",
                  "Router ospf 1",
                  "Network __merega_1_ __back_maska_1_ area 0",
                  "Network __merega_4_ __back_maska_4_ area 1",
                  "Network __merega_3_ __back_maska_3_ area 0",
                  "Network __merega_2_ __back_maska_2_ area 0",
                  "exit",
```

```
            "exit",
            "wr"
        ],
        'ldp': [
            "Enable",
            "Conf term",
            "ip cef",
            "mpls label protocol ldp",
            "mpls ldp router-id loopback 0",
            "int fa 0/0",
            "mpls ip",
            "mpls mtu 1512",
            "exit",
            "int fa 1/0",
            "mpls ip",
            "mpls mtu 1512",
            "exit",
            "exit",
            "wr"
            ]
        },
    "Rout2": {
        'ospf': [
            "Enable",
            "Conf term",
            "hostname Router_2",
            "int loopback 0",
            "ip add  __adds_3_  __mask_3_",
            "Interface fa 0/0",
            "Ip add __adds_1_ __mask_1_",
            "No shutdown",
            "Interface fa 1/0",
            "Ip add __adds_2_ __mask_2_",
            "No shutdown",
            "Router ospf 1",
            "Network __merega_1_ __back_maska_1_ area 0",
            "Network __merega_3_ __back_maska_3_ area 1",
            "Network __merega_2_ __back_maska_2_ area 0",
            "exit",
            "exit",
            "wr"
        ],
        'ldp': [
            "Enable",
            "Conf term",
            "ip cef",
            "mpls label protocol ldp",
            "mpls ldp router-id loopback 0",
            "int fa 0/0",
            "mpls ip",
            "mpls mtu 1512",
            "exit",
            "int fa 1/0",
            "mpls ip",
            "mpls mtu 1512",
            "exit",
            "exit",
            "wr"
```

```
            ]
      },
"Rout3": {
      'ospf': [
            "Enable",
            "Conf term",
            "hostname Router_3",
            "int loopback 0",
            "ip add __adds_3_   __mask_3_",
            "Interface fa 0/0",
            "Ip add __adds_1_ __mask_1_",
            "No shutdown",
            "Interface fa 1/0",
            "Ip add __adds_2_ __mask_2_",
            "No shutdown",
            "Router ospf 1",
            "Network __merega_1_ __back_maska_1_ area 0",
            "Network __merega_2_ __back_maska_2_ area 0",
            "Network __merega_3_ __back_maska_3_ area 1",
            "exit",
            "exit",
            "wr"
      ],
      'ldp': [
          "Enable",
            "Conf term",
            "ip cef",
            "mpls label protocol ldp",
            "mpls ldp router-id loopback 0",
            "int fa 0/0",
            "mpls ip",
            "mpls mtu 1512",
            "exit",
            "int fa 1/0",
            "mpls ip",
            "mpls mtu 1512",
            "exit",
            "exit",
            "wr"
            ]

      },
"Rout4": {
      'ospf': [
            "Enable",
            "Conf term",
            "hostname Router_4",
            "int loopback 0",
            "ip add  __adds_4_   __mask_4_",
            "Interface fa 0/0",
            "Ip add __adds_1_ __mask_1_",
            "No shutdown",
            "Interface fa 1/0",
            "Ip add __adds_2_ __mask_2_",
            "No shutdown",
            "Interface fa 2/0",
            "ip add __adds_3_ __mask_3_",
            "No shutdown",
```

```
                "exit",
                "Router ospf 1",
                "Network __merega_1_ __back_maska_1_ area 0",
                "Network __merega_4_ __back_maska_4_ area 1",
                "Network __merega_3_ __back_maska_3_ area 0",
                "Network __merega_2_ __back_maska_2_ area 0",
                "exit",
                "exit",
                "wr"
            ],
            'ldp': [
                "Enable",
                "Conf term",
                "ip cef",
                "mpls label protocol ldp",
                "mpls ldp router-id loopback 0",
                "int fa 0/0",
                "mpls ip",
                "mpls mtu 1512",
                "exit",
                "int fa 1/0",
                "mpls ip",
                "mpls mtu 1512",
                "exit",
                "exit",
                "wr"
                ]


            }
}

function generateCode(){
     var data = inputData()
     var dataIndex = -1
     var netType = $('[name="merega-type"]:checked').val()
     var tabCodes = codes[$('.block.active')[0].className.split('
')[1]][netType]

     var codesWithData = ""
     for (var i in tabCodes){
          var val = tabCodes[i]

                val = val.replace('__adds_1_', data[0].adds)
                val = val.replace('__adds_2_', data[1].adds)
                val = val.replace('__merega_1_', get-
merega(data[0].adds, data[0].maska))
                val = val.replace('__merega_2_', get-
merega(data[1].adds, data[1].maska))
                val = val.replace('__mask_1_', data[0].maska)
                val = val.replace('__mask_2_', data[1].maska)
                val = val.replace('__back_maska_1_', invertMas-
ka(data[0].maska))
                val = val.replace('__back_maska_2_', invertMas-
ka(data[1].maska))

                val = val.replace('__adds_net_1_', get-
addsmerega(data[0].adds, data[0].maska))
```

```
                val = val.replace('__adds_net_2_', get-
addsmerega(data[1].adds, data[1].maska))
                val = val.replace('__adds_3_', data[2].adds)
                     val = val.replace('__merega_3_', get-
merega(data[2].adds, data[2].maska))
                     val = val.replace('__mask_3_', data[2].maska)
                     val = val.replace('__back_maska_3_', invertMas-
ka(data[2].maska))
                     val = val.replace('__adds_net_3_', get-
addsmerega(data[2].adds, data[2].maska))
                if (data[3]){

                val = val.replace('__adds_4_', data[3].adds)
                     val = val.replace('__merega_4_', get-
merega(data[3].adds, data[3].maska))
                     val = val.replace('__mask_4_', data[3].maska)
                     val = val.replace('__back_maska_4_', invertMas-
ka(data[3].maska))
                     val = val.replace('__adds_net_4_', get-
addsmerega(data[3].adds, data[3].maska))

                }

           codesWithData +=val + '<br>'
     }

     $('.code')[0].innerHTML = codesWithData
     $('#code')[0].value = codesWithData.replace(/<br>/g, '\n')
}

function getmerega(adds, maska){
     var result = ""

     var addsNumbers = adds.split('.')
     var maskNumbers = maska.split('.')

     for (var i in addsNumbers){
          var val = addsNumbers[i]

          if (maskNumbers[i] === "0"){
               val = "0"
          }
          if (i != 3){
               val += '.'
          }
          result += val
     }
     return result
}

function invertMaska(maska){
 var list = maska.split('.')
 for (var i in list){
  if (list[i] == '0'){
   list[i] = '255'
  } else if (list[i] == '255'){
   list[i] = '0'
  }
```

```
 }
 return list.join('.')
}

function getaddsmerega(adds, maska){
      var addsList = adds.split('.')
      var maskList = maska.split('.')

      for (var i in maskList){
            if (maskList[i] == '0'){
                  addsList[i] = '0'
            }
      }

      return addsList.join('.')
}
</script>
</body>
</html>
```

Додаток Б

```
.block {
display: none;
}

.topology {
    position: absolute;
     left: 100px;
     top: 60px;
     box-shadow: 0 0 0 4px rgba(173, 220, 221, 0.17), 2px 1px 6px 4px
rgba(10, 10, 0, 0.5);
    border-radius: 25px;
}


.blow .marsh{
     display: none;
}

.marsh{
         width: 70px;
          position: absolute;
          cursor: pointer;
}

.add-lab {
     float: left;
    font-size: 10pt;
    width: 40px;
    text-transform: capitalize;
    padding-top: 7px;
     font-family: "Times New Roman";
     margin-left: 10px;
}

.add-m{
     position: absolute;
    background: rgb(255,212,3) linear-gradient(rgb(255,212,3),
rgb(248,157,23));
    box-shadow: 0 0 0 4px rgba(173, 220, 221, 0.17), 2px 1px 6px 4px
rgba(10, 10, 0, 0.5);
     border-radius: 5px;
}

.add-m input {
     margin: 5px;
     width: 109px;
    padding: 0 3px;
    color: #000;
    font-size: 13px;
    border-radius: 3px;
}

.gen-code-b {
    margin-top: 287px;
    margin-left: 690px;
```

```
    text-align: left;
     display: none;
}

.code {
     background: #f0d8d8;
    width: 440px;
    border-radius: 5px;
    border: 3px solid;
    padding: 10px;
     overflow: auto;
     height: 210px;
    border-color: #ff69b4;

}

.Rout1 .add-m-1{
    top: 210px;
    left: 320px;
}
.Rout1 .add-m-2{
    top: 350px;
    left: 320px;
}
.Rout1 .add-m-3{
    top: 240px;
    left: 150px;
}
.Rout1 .add-m-4{
    top: 140px;
    left: 240px;
}

.Rout2 .add-m-1{
    top: 210px;
    left: 330px;
}
.Rout2 .add-m-2{
    top: 210px;
    left:  560px;
}
.Rout2 .add-m-3{
     top: 115px;
    left: 445px;
}

.Rout3 .add-m-1{
    top: 310px;
    left: 330px;
}
.Rout3 .add-m-2{
    top: 310px;
    left: 550px;
}
.Rout3 .add-m-3{
    top: 440px;
    left: 440px;
}
```

```
.Rout4 .add-m-1{
    top: 205px;
    left:  550px;
}
.Rout4 .add-m-2{
    top: 320px;
    left:  550px;
}
.Rout4 .add-m-3{
    top: 205px;
    left: 740px;}

.Rout4 .add-m-4{
    top: 130px;
    left: 645px;}

img.marsh.Rout1 {
     left: 300px;
    top: 285px;
}

img.marsh.Rout2 {
    left: 488px;
    top: 180px;
     height: 48px;
}

img.marsh.Rout3 {
    left: 483px;
    top: 368px;
}

img.marsh.Rout4 {
    left: 687px;
    top: 269px;
}

.button {
top: 120px;
  position: relative;
  display: inline-block;
  width: 10em;
  height: 2.5em;
  line-height: 2.5em;
  vertical-align: middle;
  text-align: center;
  text-decoration: none;
  text-shadow: 0 -1px 1px #777;
  left: 1060px;
  color: black;
  outline: none;
    margin: 10px 5px;
  border: 1px solid rgb(250,172,17);
  border-radius: 5px;
  box-shadow: inset 0 -2px 1px rgba(0,0,0,0), inset 0 1px 2px
rgba(0,0,0,0), inset 0 0 0 60px rgba(255,255,0,0);
  background:  rgb(255,212,3) linear-gradient(rgb(255,212,3),
```

```
rgb(248,157,23));
}
.button:active {
top: 120px;
left: 17px;
  box-shadow: 0 0 0 60px rgba(0,0,0,.05) inset;
  left: 1060px;
}

.radio-buttons{
    margin-left: 30px;
     }

input[type="radio"] {

  opacity: 0;}

.radio-buttons label {
  top: 150px;
  position: relative;
  display: inline-block;
  padding-left: 30px;
  padding-right: 10px;
  line-height: 36px;
  cursor: pointer;
  left: 1020px;
}
.radio-buttons label::before {
  border-radius: 6px;
  content: " ";
  position: absolute;
  top: 6px;
  left: 0;
  display: block;
  width: 24px;
  height: 24px;
  border: 2px solid rgb(250,172,17);
  z-index: -1;
}

.radio-buttons input[type="radio"]:checked + label::before {
  top: 0;
  width: 100%;
  height: 100%;
  background: rgb(255,212,3) linear-gradient(rgb(255,212,3),
rgb(248,157,23));

}
```