МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Графічний інтерфейс налаштування технології Carrier Ethernet»

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

Керівник роботи — **Великодний Д.В.**

Студентки групи ІК.мз-91с — **Долгополової О.О.**

СУМИ 2020

ЗМІСТ

ВСТУП

Розвиток комп’ютерних технологій пришвидшується з кожним роком. Багато розробок, що 10 років тому вважалися новаторським, зараз вже не справляються із стрімко зростаючим попитом користувача на якісні послуги.

Компанії хочуть мати велику надійну мережу, яка буде зв’язувати всі офіси в різних куточках країни і світу. При цьому мережа повинна бути надійною, швидкою та мати багато інших додаткових функцій. Крім того вона повинна легко налаштовуватися і ціна її повинна бути доступна.

Для задоволення всих цих потреб еволюція мережевих технологій призвела до появи технології мультипротокольної комутації або MPLS.

MPLS – це новий стандарт світу комп’ютерно-інформаційних технологій, основою якого є комутація за допомогою міток. Так, це дійсно та технологія, що дає багато можливостей для користувачів. Але для того щоб з нею працювати, необхідно мати певні знання і досвід в налаштуванні мереж.

Готовим вирішенням цієї проблеми можуть бути графічні інтерфейси, які допоможуть отримати всі необхідні налаштування, затративши при цьому мінімальну кількість зусиль. Достатньо буде лише ввести параметри мережі в створену програму і як результат ми отримаємо налаштування того чи іншого обладнання.

Ці налаштування потім можна з легкістю використати в емуляторі або на реальному обладнанні і отримати повноцінну працюючу мережу. Розробка такого інтерфейсу для налаштування мережі технології Carrier Ethernet і є основною метою даної роботи.

РОЗДІЛ 1

МЕРЕЖІ ОПЕРАТОРСЬКОГО КЛАСУ НА ОСНОВІ ТЕХНОЛОГІЇ CARRIER ETHERNET

1.1 Технологія MPLS

Технологія мультипротокольної комутації за допомогою міток (MultiProtocol Label Switching, MPLS) на сьогодні вважається однією із самих перспективних транспортних технологій. Ця технологія поєднує техніку віртуальних каналів із функціональністю стека TCP/IP. Це поєднання відбувається завдяки тому що один і той самий мережевий пристрій LSR (Label Switch Router або комутуючий по мітках маршрутизатор), виконує функції як IP-маршрутизатора, так і комутатора віртуальних каналів. При цьому це не механічне поєднання двох пристроїв, а тісна інтеграція, коли функції кожного приладу доповнюють один одного і використовуються сумісно [1].

Технологія MPLS була розроблена компанією Cisco в 2001 році для магістральних та локальних мереж Ethernet. А сьогодні ми бачимо як MPLS впроваджується в найбільш популярні направлення нових поколінь мережевих технологій [2].

Ця технологія розроблялася як спосіб побудови високошвидкісних IP-магістралей, проте область її використання не обмежується лише IP-протоколом, але й поширюється на трафік будь-якого маршрутизованого мережного протоколу. Вона має практично необмежені можливості масштабування, підвищену швидкість обробки трафіку і безпрецедентну гнучкість з точки зору організації додаткових сервісів [3].

Наступний рисунок ілюструє архітектуру MPLS-мережі, яка взаємодіє з декількома звичайними IP-мережами, які можуть навіть не підтримувати цю технологію.

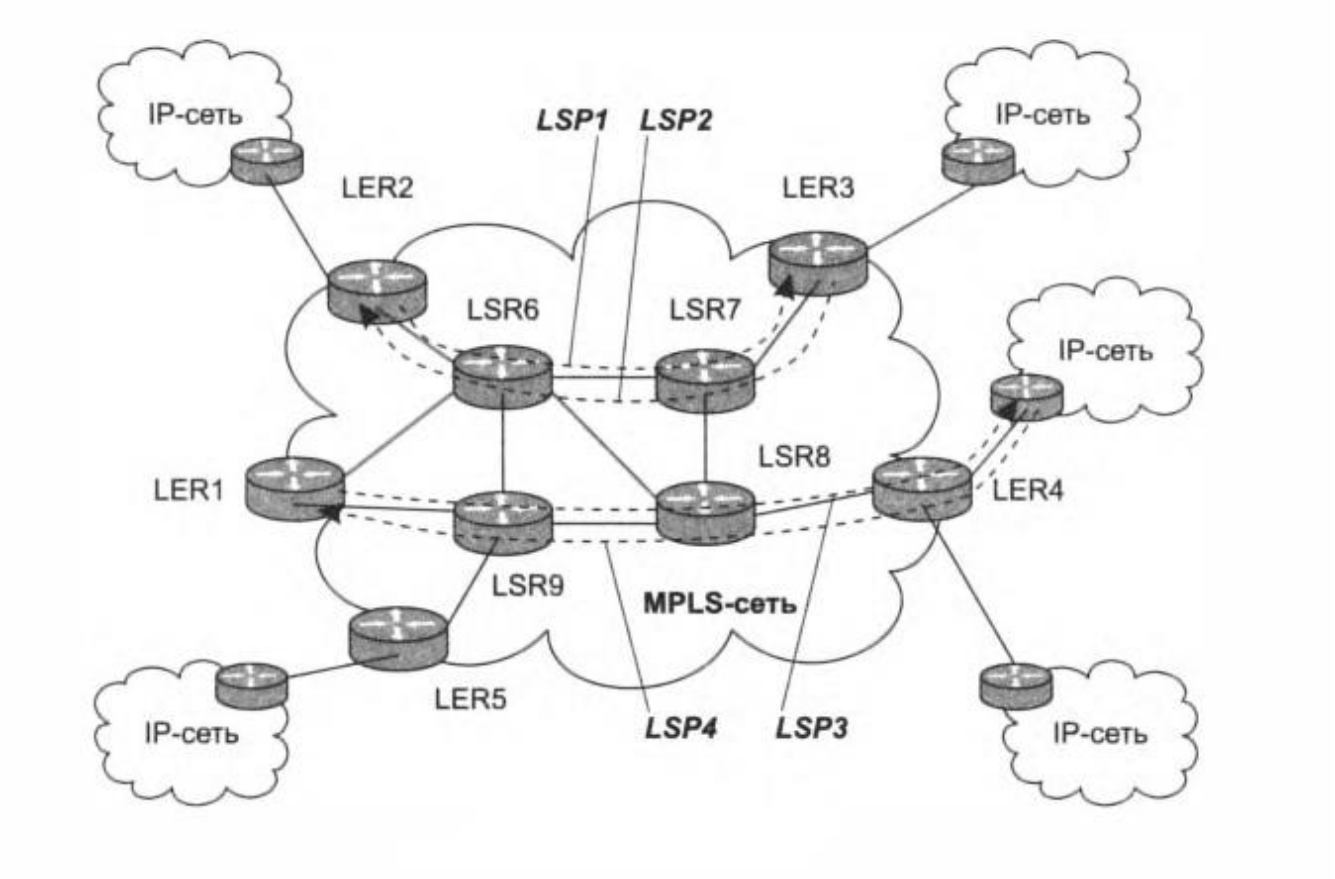

Рис. 1.1. Архітектура MPLS-мережі [1]

В даній архітектурі присутні пристрої LER (Label switch Edge Router), які завдяки своїм функціональним особливостям приймають трафік від інших мереж у вигляді стандартних IP-пакетів, а потім додають до них мітку та відправляють вздовж необхідного шляху до вихідного пристрою LSR. Слід зауважити що пакет рухається не на основі IP-адреса призначення, а на основі мітки [1]. Це допомагає маршрутизаторам точно розуміти куди потрібно направити дані, не аналізуючи знову і знову пакет с даними.

Щоб зрозуміти принцип роботи MPLS-технології, слід відмітити, що в традиційній IP-мережі кожний маршрутизатор повинен перевіряти IP шляхом постійного його пошуку в таблицях з пакетами даних і далі відправляти на наступний рівень, доки пакети не досягнуть свого пункта призначення.

MPLS технологія призначає мітку усім IP-пакетам і маршрутизатори завдяки потрібному значенню мітки приймають рішення про передачу пакета на наступний пристрій. Мітка додається до MPLS заголовку, який знаходиться між заголовком кадру (другий рівень OSI) та заголовком пакету (третій рівень OSI) [4].

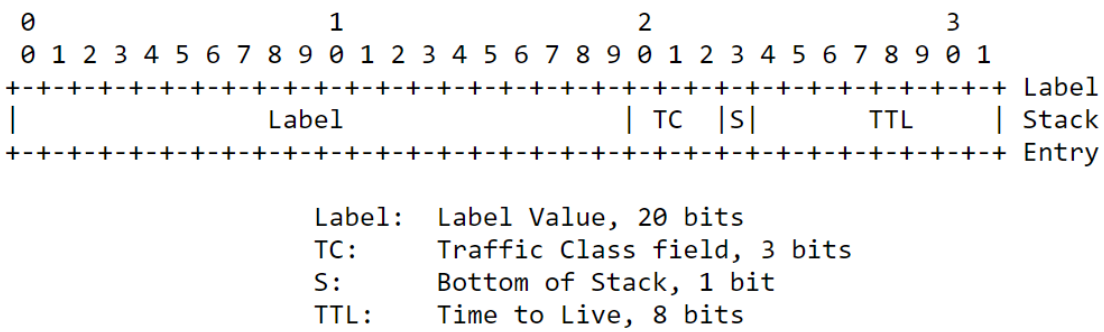

Рис. 1.2. Класичний формат мітки MPLS за регламентом RFC [5]

Мітка має певний розмір, що дорівнює 4 байтам. При цьому сам ідентифікатор (Label Value) займає лише перші 20 біт, а наступні 3 біта (Traffic Class) зарезервовані для експерементального використання. Останній біт третього байта (Bottom of Stack) використовується для вказування на закінчення стеку міток, а у четвертому байті (Time to Live) вказано час, протягом якого пакет буде існувати в мережі TTL [6].

Пакети, що рухаються за одним маршрутом, поєднуються у групи (класи) FEC (Forwarding Equivalence Class). Подібне об’днання значно знижує затрати ресурсів на обробку пакетів тому що маршрутизатору достатньо проаналізувати лише перший пакет однієї групи.

Часто мітки пакетів об’єднуються послідовно в один стек. Маршрутизатор LSR працює лиише з першою міткою в стеку до тих пір, поки пакет не досягне свого пункта призначення, де ця мітка видаляється. Якщо після цього в стеку ще залишаються мітки, кожна з них буде проаналізована та оброблена таким же чином – до тих пір, поки не буде видалена остання. При необхідності порядок міток в стеку може змінюватися, а кожна нова мітка окремо може бути замінена новою [7].

Доставка пакетів у MPLS принципово відрізняється від стандартного метода IP, що базується на адресах пунктів призначення. Замість цього використовується мітка, що розміщується у заголовку пакета.

Маршрутизатор MPLS перевіряє вхідну мітку і змінює її на відповідну вихідну мітку, перевіряючи при цьому таблицю LFIB (Label Forwarding Information Base). Розмір цієї таблиці значно менший у порівнянні з IP- таблицею маршрутизації IP-мережі.

Оскільки в мережі MPLS алгоритм роботи з міткам визначений заздалегідь, кожний пристрій працює на основі відповідних правил та не може на свій розсуд трактувати пріоритет та призначення мітки [2].

Можливості архітектури MPLS:

• забеспечення якості обслуговування за рахунок фіксованої пропускної спроможності, що особливо важливо для аудіо- та відеотрафіку;

• управління характеристиками затримки та гарантування пропускної спроможності для передачі голосу;

• конфігурування різних рівнів якості обслуговування для різних користувачів;

• ефективний механізм підтримки VPN, що полягає у відокремленні різних трафіків у межах об’єднанної мережі та забезпечення при цьому продуктивності та безпеки;

• інжиніринг трафіку TE (Traffic engineering), тобто здатність динамічно вибирати маршрути, планувати ресурси та оптимізувати використання мережі [3].

Крім того, при використанні MPLS зникає необхідність у додатковому шифруванні та інших підвищених запобіжних заходах. В таких мережах можуть передаватися будь-які дані, оскільки зміст пакету залишається незмінним впродовж всього шляху – змінюються лише мітки [6].

1.2 Організація VPN на прикладі Carrier Ethernet over MPLS (L2VPN)

З розвитком комунікаційних технологій віртуальні приватні мережі MPLS (MPLS VPN) набули широкого розповсюдження. Та навколо MPLS VPN постійно ведуться суперечки стосовно їх переваг та недоліків. З одного боку – висока ціна за розгортання та підтримку такої мережі, яку можуть дозволити собі не всі компанії. З іншого боку – безліч переваг, які отримує користувач.

Серед основних переваг MPLS VPN слід відмітити:

1. Масштабованість. MPLS була розроблена в тому числі для ефективного вирішення проблем розширення мереж, оскільки постійно зростаючі потреби користувачів вимагають регулярного розширення мереж користування.
2. Безпека. VPN-мережі MPLS забезпечують надійний рівень безпеки, що не поступається іншим технологіям. В середині магістралі дані окремих VPN-мереж переміщюються виключно окремо. А пакети кожної з них потрапляють лише в певну VPN-мережу, для якої вони і призначені.
3. Гнучка адресація. Це дозволяє користувачам використовувати власні IP-адреси, уникати використання NAT та не витрачати час на перетворення відкритих IP-адрес для користування у внутрішній мережі.
4. Перерозподіл потоків. Маршрутизація з перерозподілом потоків і резервуванням ресурсів (Traffic Engineering Routing with Resource

Reservation) дозволяє максимально виокристовувати всі ресурси мережі. Такий підхід дає можливість примусово направляти потік даних згідно заданих маршрутів. Це забезпечує захист та швидке відновлення мережі у разі відмови обладнання. Крім того такий підхід дозволяє рівномірно завантажувати мережеві канали, що також позитивно впливає на якість роботи [7].

Технологія MPLS VPN використовує віртуальний маршрутзатор (Virtual Routing and Forwarding instance, VRF), що дозволяє створити багато віртуальних маршрутизаторів на одному фізичному пристрої. Кожний віртуальний маршрутизатор відокремлений від фізичного, але одночасно і закріплений за ним та не може використовуватися іншими. Це дозволяє створювати віртуальні мережі, які не перетинаються між собою [2].

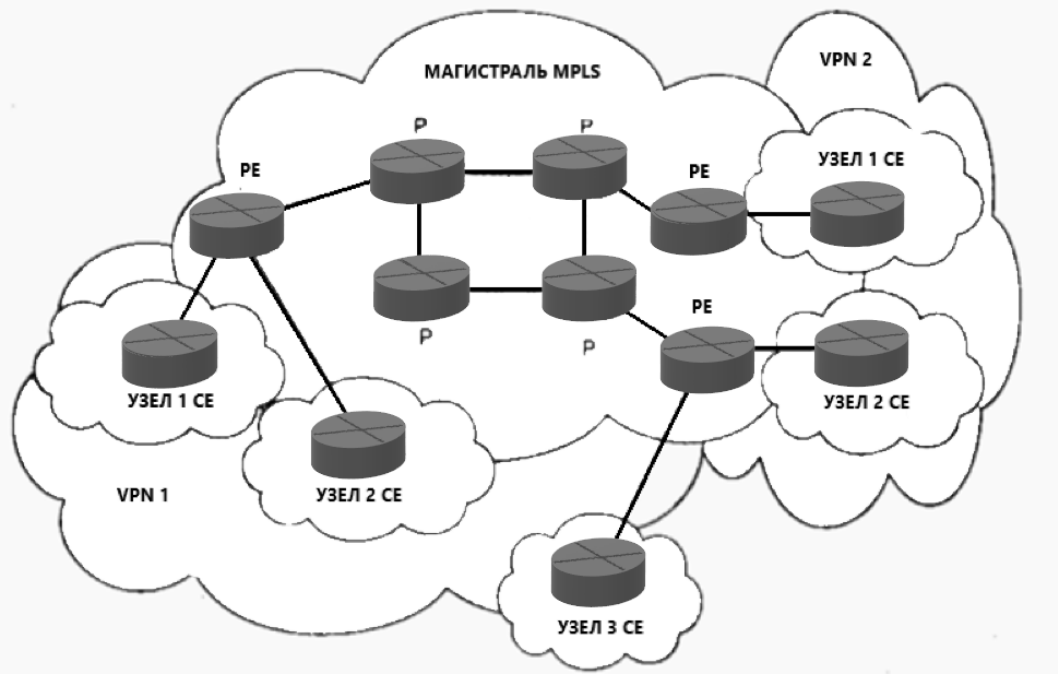

Рис. 1.3. Приклад VPN MPLS мережі [7]

Нижче розглянемо основні компоненти, що входять до складу такої мережі.

Базовий маршрутизатор MPLS (P, Provider router). Це транзитний маршрутизатор, який лише комутується по транспортній мітці. Він не має інтерфейсів, що прив’язані до VPN, не містить маршрут VPN-мережі та ніколи не під'єднується безпосередньо до маршрутизатора користувача.

Граничний маршрутизатор мережі MPLS (Provider Edge router – PE router або PER). Підтримує інтерфейс та з’єднується з P-маршрутизатором та іншими CE- і PE-маршрутизаторами. Містять VPN-маршрути для тих VPN-мереж, які підтримують.

Граничний маршрутизатор користувача (Customer Edge router – CE router або CER). Може не підтримувати функції MPLS, а для підтримки з’єднання використовують звичайні методи маршрутизації. Ніколи не з’єднуються прямим зва’язком з P-маршрутизаторами.

Маршрутизатор користувача (Customer router – C-router). Це внутрішній маршрутизатор клієнта. Може не підтримувати функції MPLS, а використовувати традиційні методи маршрутизації [7].

На сьогодні існує дві різні технології MPLS VPN, що базуються на способі передачі пакетів. В L2VPN пакети передаються через другий рівень моделі OSI. В L3VPN відповідно через третій.

L3VPN MPLS в цій роботі ми не будемо детально розглядати, але декілька слів слід сказати. Дана технологія використовує для передачі пакетів через мережевий рівень лише протокол IP. Одна із основних переваг даної технології – ізоляція трафіку користувачів на магістральній мережі та гарантія того, что дані різних груп користувачів не будуть змішуватися. Серед недоліків слід відмітити неможливість працювати з іншими протоколами, що реалізуються через канальний рівень [2].

В даній роботі ми зосередимось на технології L2VPN MPLS. Чому саме на ній? Тому що саме її сьогодні можна побачити у більшості провайдерських мереж. І цьому є певне обґрунтування.

По-перше, це особливість маршрутизаторів, які передають MPLS-пакети, не зважати на зміст пакета і не аналізувати його, але при цьому чітко розрізняти трафіки різних сервісів.

По-друге, важливою є функція AToM (Any Transport over MPLS), яка дозволяє інкапсулювати трафік будь-якого канального рівня в MPLS-пакет. Функція AtoM підтримує протоколи ATM Cell Relay, Ethernet, Frame Relay, High-Level Data Link Control, PPP.

Технологія L2VPN MPLS стандартизована документами IETF і працює з протоколами другого канального рівня, з яким працюють більшість сучасних дата-центрів [2].

Наступний рисунок ілюструє зміни у стеку міток в процесі переадресації.

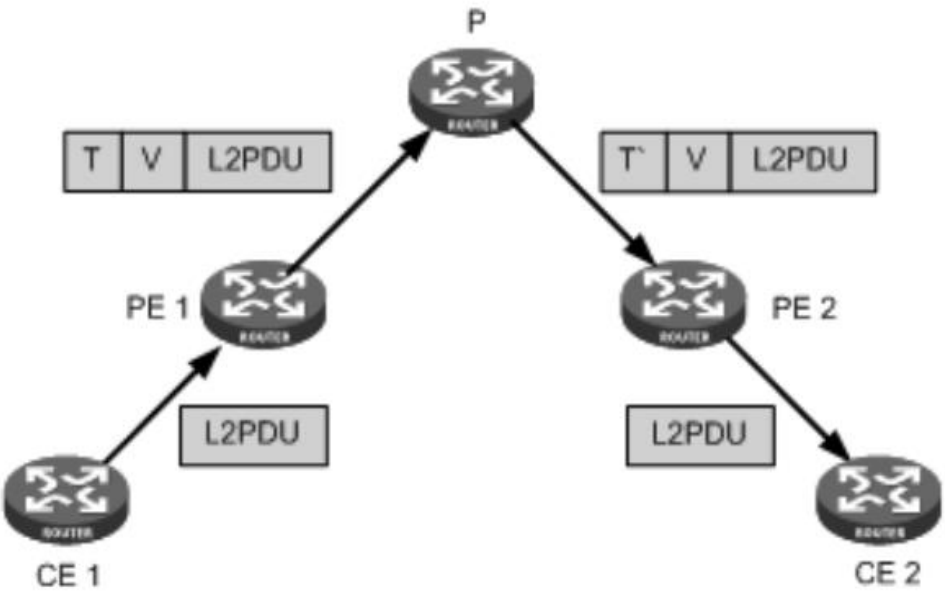

Рис. 1.4. Обробка стеку міток MPLS VPN [8]

Основні принципи роботи і обладнання такі самі як для всих MPLS VPN мереж.

Маршрутизатор користувача (CE) завжди знаходиться в клієнтській мережі. Це може бути маршрутизатор, комутатор або хост. Може мати один або кілька інтерфейсів, які безпосередньо підключені до мереж постачальника послуг. Підтримки MPLS не потребує. Про наявність VPN інформації не має.

Граничний маршрутизатор (PE) провайдера знаходиться на межі мережі провайдера і з'єднує один або декілька маршрутизаторів користувача. В мережі MPLS всі служби VPN обробляються на PE-маршрутизаторі.

Маршрутизатор провайдера (P) – основний пристрій в мережі провайдера. Він не зв'язується з маршрутизатором користувача, а має лише можливість переадресації MPLS.

Мітки, що використовуються для передачі пакетів в мережі MPLS:

• L2PDU який є предметом транспорту (трафік, отриманий від маршрутизатора користувача);

• зовнішня мітка або тунельна мітка (Outer label) використовується для передачі пакетів з одного PE пристрою на інший;

• внутрішня мітка або VC мітка (Inner label) використовується для ідентифікації різних зв'язків між VPN.

Аналізуючи ці мітки, маршрутизатор провайдера розпізнає до якого маршрутизатора користувача слід направити пакет [8].

Користувачам пропонується два типа послуг Ethernet-мереж для організації VPN на другому рівні моделі OSI і на базі технології MPLS. Це VPWS (Virtual Private Wire Service) та VPLS (Virtual Private LAN Service). Ці віртуальні мережі базуються на основі побудови псевдоканалів (pseudowire), які зв’язують граничні PE-маршрутизатори мережі провайдера [9].

Різні типи технологій, які виділяються під MPLS VPN, можна побачити на наступній схемі.

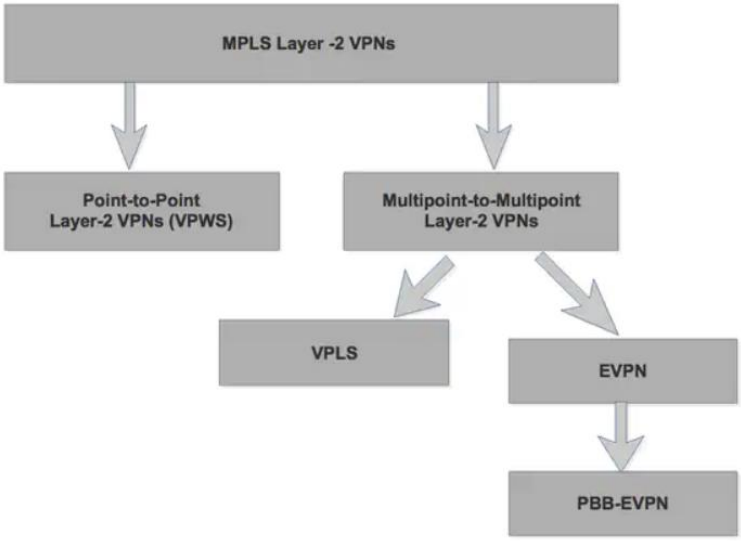

Рис. 1.5. - Модель MPLS L2VPN [1]

Point-to-Point або VPWS (Virtual private Wire Service) підтримує будь-які типи протоколів канального рівня. В основі лежить концепція PW (Pseudo Wire або псевдопровід). Якщо вам потрібно з'єднати дві точки одна з іншою, мережа провайдера буде для вас як один віртуальний кабель.

Ця технологія досить проста та зрозуміла в плані організації, передачі даних та роботи службових протоколів.

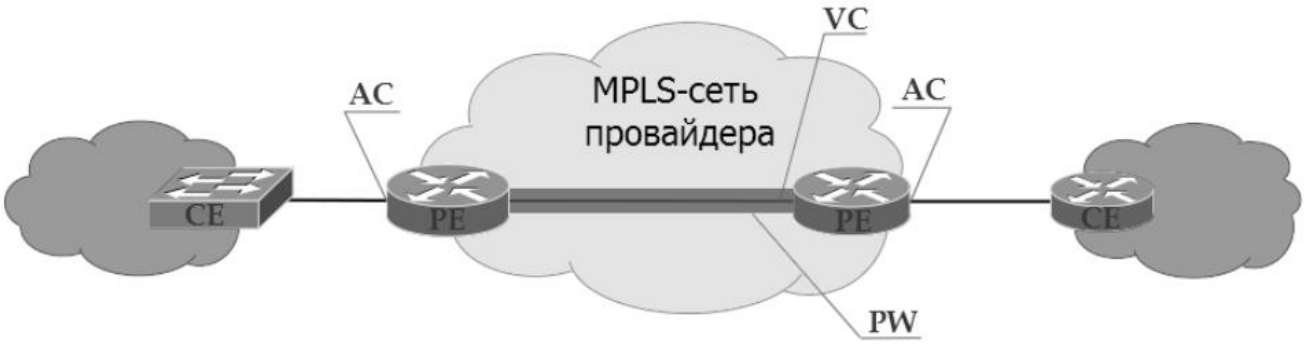

Рис. 1.6. Приклад VPWS мережі [10]

PE (Provider Edge) – граничний маршрутизатор провайдера

CE (Customer Edge) – маршрутизатор користувача

AC (Attached Circuit) – інтерфейс на PE для підключення користувача

VC (Virtual Circuit) – віртуальне однонаправлене з'єднання крізь загальну мережу, що імітує оригінальне середовище для клієнта. З'єднує між собою AC-інтерфейси різних провайдерських маршрутизаторів в єдиний канал AC–>VC–>AC.

PW (Pseudo Wire) – віртуальний двонаправлений канал передачі даних між двома PE і складається із двох однонапрвлених VC [10].

Для організації Point-to-Point схеми існує технологія AToM (Any Transport Over MPLS). Ця інтегральна технологія забезпечує передачу L2 фреймів через MPLS мережу та представлена frame Relay over MPLS, Ethernet over MPLS та ATM over MPLS.

AToM використовує LDP сесії між граничними провайдерськими маршрутизаторами для встановлення та підтримки з'єднання. Передача пакетів відбувається за допомогою міток Top Label або Tunnel Label (з'єднує граничні маршрутизатори) та Bottom Label або VC Label (визначає інтерфейс VPN клієнта на PE маршрутизаторі).

EoMPLS інкапсулює Ethernet фрейми у MPLS-пакети та, використовуючи стеки міток, транспортує їх крізь MPLS мережу [11].

На малюнку, що наведено нижче, розглянуто як відбувається передача даних при використанні VPWS. Але спочатку декілька слів слід сказати про мітки, що допоможе в розумінні роботи даної технології.

Мітка (label) – є частиною заголовку MPLS. Спираючись на неї, маршрутизатор приймає рішення що робити з пакетом та яку мітку слід йому присвоїти. Мітка може приймати значення від 0 до 1048575.

Стек міток (Label Stack) – сукупність міток одного пакету. Таких міток може бути різна кількість для одного пакету і об’єднуються вони в стек. Верхня мітка в стеку – основа для прийняття рішення маршрутизатору що далі робити з пакетом.

Push мітка (Push Label) – це процес додавання мітки до пакета даних. Ця операція відбувається на першому маршрутизаторі в мережі MPLS.

Swap мітка (Swap Label) – процедура заміни мітки, що відбувається на проміжних маршрутизаторах в мережі MPLS. Маршрутизатор отримує пакет з певною міткою, аналізує її та відправляє його далі вже з іншою міткою.

Pop мітка (Pop Label) – процедура видалення мітки. Відбувається на останньому маршрутизаторі, який отримує пакет MPLS та видаляє верхню мітку перед відправленням пакету далі [12].

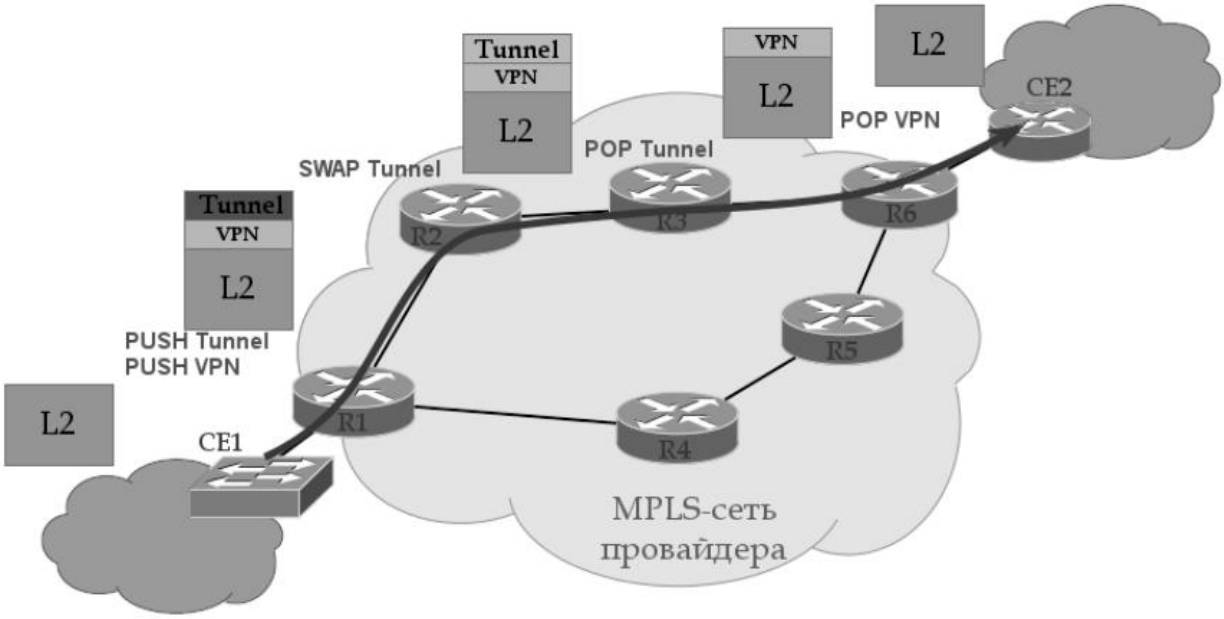

Рис. 1.7. Передача даних у VPWS [10]

Як же відбувається безпосередньо передача пакету крізь VPWS?

Між R1 та R6 вже існує транспортний LSP і R1 вже знає значення транспортної (тунельної) мітки і вихідний інтерфейс R6.

R1 має вже налаштовані AC-інтрефейси, які закріплені за певним ідентифікатором клієнта (VC ID).

Від маршрутизатора користувача (CE1) на R1 надходить кадр і тут він отримує сервісну мітку (VPN-мітка), яка збережеться до кінця шляху незмінною. Це внутрішня мітка в стеку.

R1 маршрутизатору відома IP-адреса віддаленого PE-маршрутизатору. На основі цього він визначає транспортну мітку та додає її до стеку міток MPLS. Ця транспортна мітка буде змінюватися під час транспорту пакета в середині MPLS мережі [10].

На передостанньому мершрутизаторі ця транспортна мітка видаляється і відбувається оптимізація стеку (PHP, Penultimate Hop Popping). Це необхідно для того щоб останній маршрутизатор виконував не 2 lookup (мітка та адреса), а лише один (адреса) [13].

На останню точку R6 пакет приходить вже лише із сервісною VPN-міткою, яка аналізується і приймається рішення на який інтерфейс передати розпакований кадр [10].

Point-to-Multipoint або Virtual Private LAN Service.

VPLS або Virtual Private LAN Service. Ця технологія використовується коли клієнт має декілька або багато точок доступу і всі вони повинні між собою взаємодіяти та обмінюватися даними. Вона дозволяє поєднати розподілені локальні мережі в одну. Принцип роботи дуже схожий зі звичайним Ethernet-комутатором, оскільки в мережу підключається декілька точок і забезпечується їх L2 взаємодія. При цьому клієнт зовсім не бачить мережу провайдера [10].

До переваг VPLS слід віднести наступні особливості:

• швидкий обмін даними всередині мережі;

• висока надійність передачі інформації;

• спільна робота з базами даних та документами;

• високоякісний трафік, що дозволяє між офісами організовувати відеоконференції.

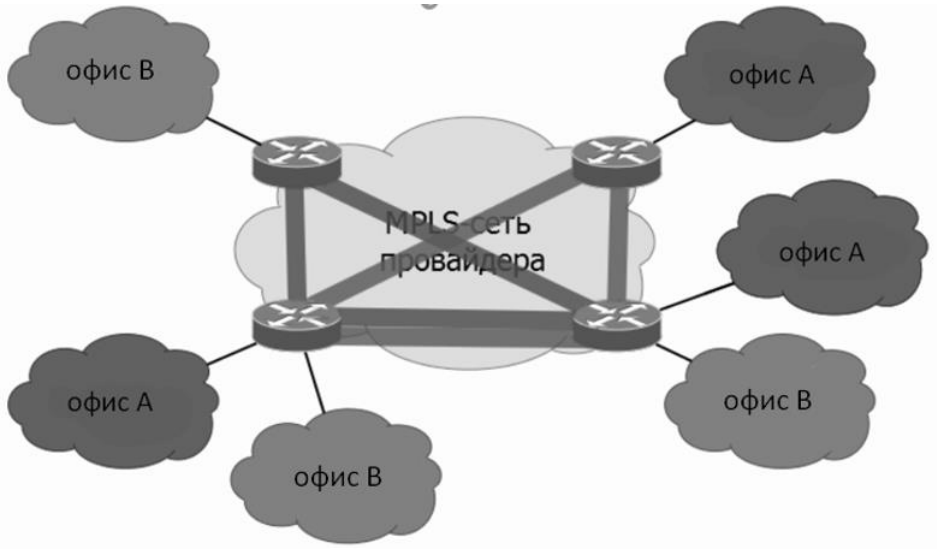

Рис. 1.8. Приклад VPLS мережі [10]

VPLS виглядає і працює як і VPWS, з тією лише відмінністю що до неї додається шаг роботи з MAC-адресами та таблицею. На наступному прикладі можемо більш детально розглянути принцип роботи VPS мережі.

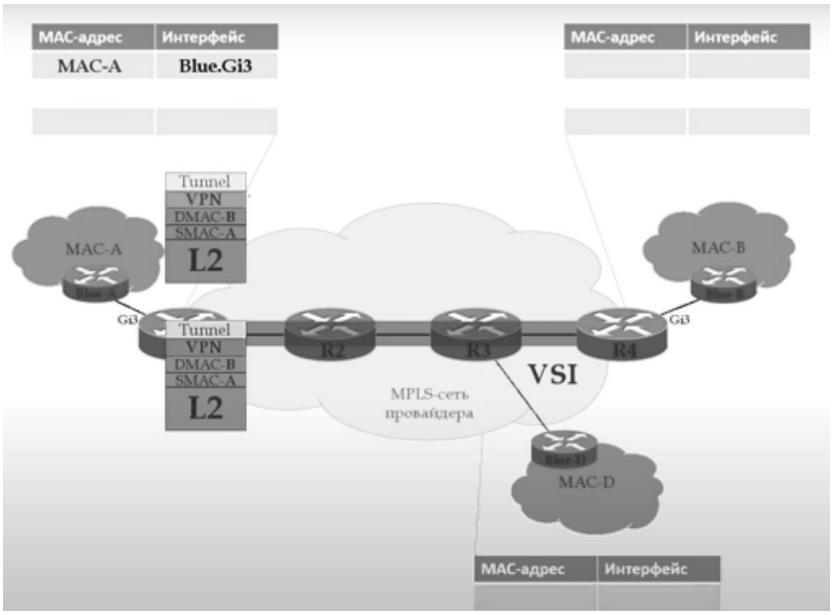

Рис. 1.9. Принцип роботи VPLS мережі [10]

Користувач відправляє Ethernet-кадр, який потрапляє на AC-порт PE-маршрутизатора.

Ingress PE вивчає MAC-адресу відправника у Ethernet-заголовку. Далі можливо два сценарія.

Якщо MAC-адреса отримувача присутня в таблиці, PE знаходить вихідний інтерфейс для неї (Pseudo wire або фізичний інтерфейс). Pseudo wire (PW) – це завжди канал між двома IP-адресами, тому дуже легко зрозуміти адресу необхідного PE, знайти для нього відповідну мітку та поставити верхньою у стеку.

Якщо MAC-адреса невідома, то PE маршрутизатор виконує широкомовну розсилку всим доступним PE. Для цього локальний PE кожному віддаленому маршрутизатору призначає мітку і додає її до кадру. Потім додається ще і транспортна мітка, причому кожному маршрутизатору своя.

Віддалений маршрутизатор при отримані кадра також може діяти за декількома сценаріями.

Якщо MAC-адреса невідома, то вона заноситься до таблиці, де в якості вихідного інтерфейсу Ingress PE буде вказаний PW.

Якщо MAC-адреса відома, то відправляє кадр в необхідний порт, який зазначений у таблиці. [10]

1.3 Стандарт IEEE 802.1Q. Технологія Q-in-Q

Q-in-Q або Double VLAN – це технологія, яка дозволяє робити подвійне тегування трафіка, призначаючи йому одразу дві мітки. В деяких джерелах також можна зустріти назву 802.1Q tunneling, яка пішла від номеру стандарту 802.1Q, де саме і описується процедура тегування трафіка для передачі по мережі інформації про належність до певного VLAN.

Суть даної технології в інкапсуляції тегів IEEE 802.1Q VLAN в теги другого рівня 802.1Q tag на граничних комутаторах провайдера PE (Provider Edge).

Провайдер додає цей тег до кожного фрейму, що надходить від клієнта з унікальним VLAN тегом. Завдяки цьому провайдер може використовувати унікальні VLAN для надання послуг клієнтам, які мають їх декілька в своїй мережі. При цьому VLAN клієнта зберігаються і трафік від різних клієнтів сегментується навіть якщо він передається в одному VLAN [14].

Найголовнішою перевагою такого рішення є легкість запровадження і використання цієї технології. Для цього не потрібно унікальне і коштовне обладнання, не потрібно використовувати складні протоколи маршрутизації між провайдером та клієнтом. З боку клієнта це виглядає як звичайне пряме з'єднання на 2 рівні.

Схематично Q-in-Q функцію можна зобразити наступним чином.

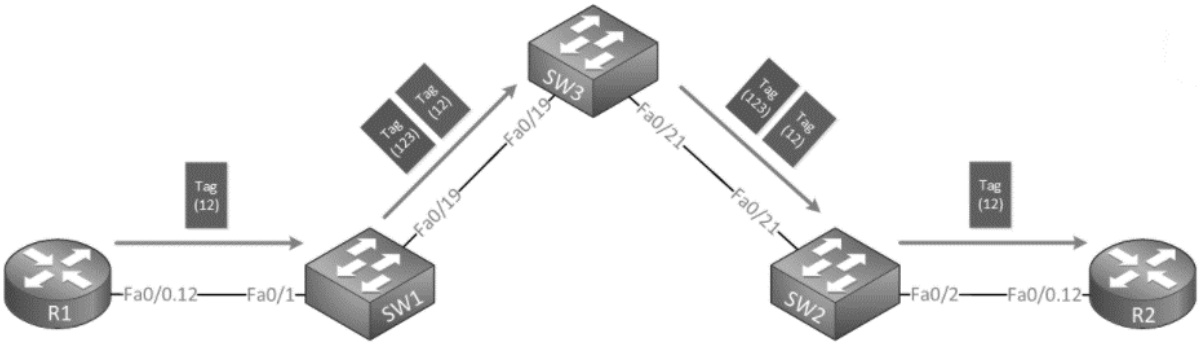

Рис. 1.10. Приклад технології Q-in-Q [15]

Припустимо, що клієнт хоче використовувати VLAN 12 для трафіку між офісами. Провайдер для цього клієнта хоче використовувати VLAN 123.

З маршрутизатору R1, що належить клієнту, вийшов трафік, позначений тегом Tag (12). Щойно пакет доясгне маршрутизатора SW1, що належить провайдеру, до нього буде додана ще одна мітка Tag (123). Перед тим як маршрутизатор SW2 буде відправляти пакет до R2, він видалить другий тег і до R1 надійде пакет лише з тегом Tag (12) [15].

При такому підході кожний з тегів додає по 4 байти до кадру Ethernet. Сам кадр буде мати наступний вигляд.

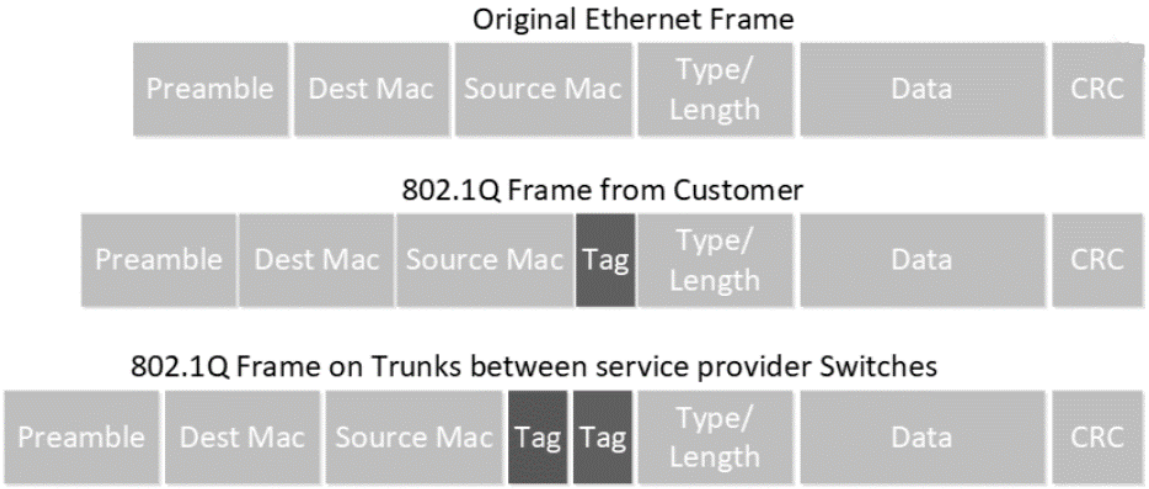

Рис. 1.11. Формат кадру Ethernet з подвійним тегом [15]

Окрім відсутності необхідності узгоджувати теги VLANів, дана технологія дає можливість значно збільшити загальну кількість VLANів.

Звичайна кількість VLAN – 4094. При використанні подвійного тегування технології Q-in-Q, ця кількість зростає до 16.760.836 (4094*4094). А це значно розришює можливості клієнтів і провайдерів.

Q-in-Q функція може бути реалізована двома способами, а саме Port-based Q-in-Q та Selective Q-in-Q. Головная відмінність між ними полягає в управлінні зовнішними тегами (Outer tag). Port-based Q-in-Q функція призначає ідентифікатор SP-VLAN, що відповідає ідентифікатору PVID порта, автоматично кожному кадру. Функція Selective Q-in-Q більш гнучка і дає можливість задавати пріоритет та додавати зовнішній тег вибірково, в залежності від внутрішнього тега (Inner tag) [16].

1.4 Стандарт IEE 802.1ah. Технологія Mac-in-Mac

Mac-in-Mac – пакетна технологія передачі трафіку, основою якої є інкапсуляція однєї MAC-адреси в іншу. Також ця технологія відома під назвою PBB (Provider Backbone Bridges). Свою назву Mac-in-Mac технологія отримала через те що має два рівні MAC-адрес.

Згідного даного стандарту адресний простір клієнта та провайдера розділяється за рахунок того, що на граничних комутаторах провайдера (BEB, Backbone Edge Bridges) відбувається інкапсуляція Ethernet кадрів. При цьому Ethernet кадр клієнта повністю інкапсулюється в нові кадри, які потім використовуються в межах мережі провайдера для транспортування кадрів клієнта до вихідного граничного комутатора. Таким чином мережа провайдера всередені представлена незалежною ієрархією MAC-адрес та VLANів. Але коли кадри знаходяться в PPB, то в якості адрес використовуються MAC-адреси граничних комутаторів (BEB, Backbone Edge Bridges).

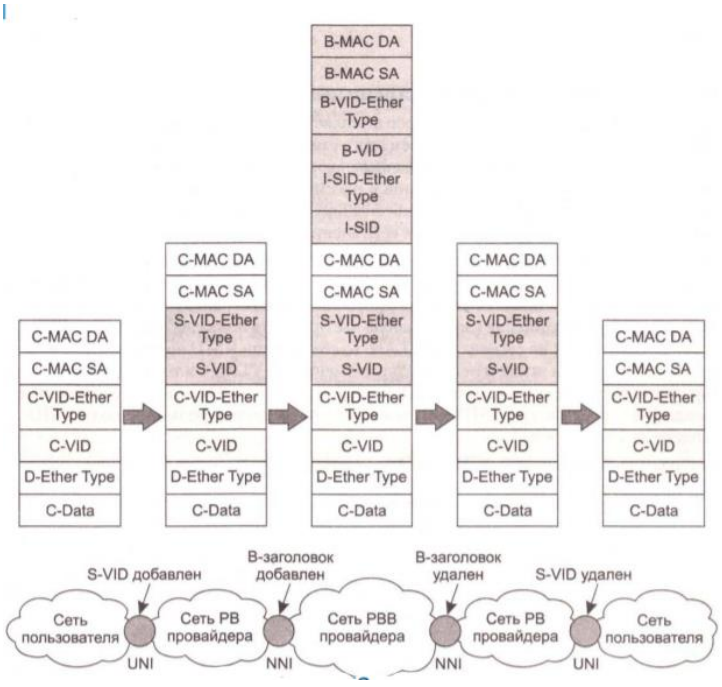

Рис. 1.12. MAC-in-MAC. Формат заголовку кадра IEE 802.1ah [17]

B-MAC DA (Backbone destination address) – магістральна адреса отримувача
B-MAC SA (Backbone source address) – магістральна адреса відправника
B-VID (Backbone VLAN tag) – магістральний тег віртуальної мережі
I-SID (Service instance tag) – тег екземпляру сервіса
C-MAC DA (Customer destination address) – адреса отримувача користувача
C-MAC SA (Customer source address) – адреса відправника користувача
S-VID (Service provider VLAN tag) – тег віртуальної мережі провайдера
C-VID (Customer VLAN tag) – тег віртуальної мережі користувача
Data (Data) - дані

Перевагою технології PBB є підвищення продуктивності магістралі за рахунок значного скорочення числа записів в адресних таблицях PBB комутаторів. При даному підході вони містять лише B-MAC-алреси. Але зростає навантаження на гарничні PBB комутатори, які повинні відображувати C-MAC та B-MAC адреси, а також інкапсуляцію.

Особливу увагу слід приділити полям I-SID та I-SID Ether type, бо саме вони були додані до кадру 802.1ah для забезпечення здатності мережі змінювати масштаб. I-SID вказує на VLAN в мережі PBB. А так як мережа PBB ділиться на сегменти B-VLAN, то з'днання I-SID – логічні сполучення всередині сегментів.

Що стосується MAC-адрес, то магістральним комутаторам мережі PBB не потрібно знати адреси клієнтів. Вони оперують лише комбінацією B-MAC/B-VID.

Граничні комутатори мають різну поведінку в залежності від типу сервіса. Якщо це E-LINE («точка-точка»), то граничні комутатори не використовують MAC-адреси клієнтів, а всі кадри передаються одному вихідному граничному комутатору.

Якщо ми говоримо про сервіс E-LAN та E-TREE («багатоточка-багатоточка»), то тут у вхідного комутатора є декілька вихідних граничних комутаторів. В таких випадках граничні комутатори, на основі вивчення MAC-

адрес клієнтів, відсилають кадр вихідному комутатору, який пов'язаний з тією мережою, де знаходиться MAC-адреса призначення C-MAC DA. Якщо ж клієнтська адреса ще не вивчена – комутатор вказує в полі B-MAC широкомовну адресу [17].

1.5 Постановка задачі

На основі зібраної та проаналізованої інформації можна сформулювати етапи виконання практичної частини кваліфікаційної магістерської роботи.

1. В симуляторі GNS3 змоделювати мережу з підтримкою технології Carrier Ethernet.
2. Проаналізувати дані за допомогою програми Wireshark. Побачити технологію транспорту пакетів за допомогою міток. Проаналізувати тип міток та принципи їх призначення та видалення.
3. Для оптимізації подальшого налаштування подібних схем необхідно розробити графічний інтрефейс налаштування Carrier Ethernet. Він допоможе налаштовувати телекомунікаційну мережу, не володіючи виключними знаннями мережевих технологій та не маючи великого досвіду в роботі з обладнанням та спеціальними програмами. Отриманий графічний інтерфейс повинен бути простим, зрозумілим і легким у використанні.
4. Перевірити налаштування, отримані за допомогою графічного інтерфейсу, в емуляторі GNS3.

РОЗДІЛ 2

МОДЕЛЮВАННЯ ТА НАЛАШТУВАННЯ L2 VPN В ЕМУЛЯТОРІ GNS3

2.1 Конфігурація мережі з використанням емулятора GNS3

Моделювання комп'ютерних мереж – популярний напрямок в розробці та дослідженні телекомунікаційних технологій. Велика кількість емуляторів було створено для побудови та налаштуванная мереж. Цей підхід дає не лише можливість вивчати мережі, будувати їх та досліджувати рух даних по мережі. Такі емулятори допомагають будувати великі мережі для компаній, які можуть бути розташовані на різних континентах.

Спроектувати мережу в емуляторі – це не одне й те саме що намалювати її в графічному редакторі. Емулятор дозволяє, перш за все, побачити масштаб мережі та все необхідне обладнання, що буде потрібне в реальному житті. Крім цього, за домопомогою таких програм можна створити декілька варіантів однієї і тієї ж мережі, проаналізувати трафік, навантаженість та відмовостійкість і вже на основі отриманих даних вибрати один із варіантів.

Програми-емулятори дозволяють передбачити і уникнути багатьох помилок. А помилки в масштабах світових компаній – це, інколи, мільойони доларів зекономлених коштів.

GNS3 або Graphical Network Simulator – одна з найпопулярніших програм, за допомогою якої можна моделювати мережі різної складності та архітектури. З'явилася вона в 2007 році завдяки дипломній роботі Джеремі Гросмана, в рамках якої він і почав її розробку. З часом ця програма розвивалась і удосконалювалась і сьогодні це, наразі, одна із найвідоміших і потужних програм-емуляторів. Це чудовий додатковий інструмент для виконання різних лабораторних робіт та завдань для інженерів мереж та адміністраторів [19].

Це безкоштовна програма з відкритим кодом, яка може бути інстальована на будь-яку операційну систему. На відміну від Cisco Packet Tracer та інших платних емуляторoів, GNS3 можна знайти в інтрнеті і завантажити собі на комп'ютер, не порушуючи ніяких законів.

Серед головних переваг слід відмітити наступні:

• майже все обладнання, що представлене у програмі, має повний функціонал і практично повністю відображає особливості і роботу реальних приладів;

• можливість побудувати мережу, використовуючи обладнання різних компаній, а не лише одного виробника. Ця особливість наближає роботу в емуляторі GNS3 до реального середовища і дозволяє побачити сумісність в роботі тих чи інших пристроїв;

• можливість побудувати мережу з використанням повноцінних робочих станцій, з певним типом операційної системи. Це дозволяє створити повноцінні налаштування на кінцевих точках, перевірити підключення VPN та роботу інтернету.

Незважаючи на всю привабливість і доступність даної програми, не можна не відмітити і ряд недоліків, що їй властиві.

Перший і основний недолік – це відсутність можливості емулювати комутатори при створенні мереж. Ця проблема виникла через те що реальні комутататори мають складну структуру, яку досить важко реалізувати в емуляторі.

Ще один важливий недолік – це високі системні вимоги для роботи з цією прогрмамою. В цьому GNS3 дещо поступається Cisco Packet Tracer. Мінімальний об'єм оперативної памяті для роботи з GNS3 – це 4Гб. Але складні комплексні мережі за таких умов побудувати все ж таки буде складно [18].

Інтерфейс постійно змінювався і вдосконалювался. Як результат програма вийшла проста та зрозуміла. Тепер всі типові функції та обладання зібрані в окремі групи, що значно полегшує роботу з емулятором.

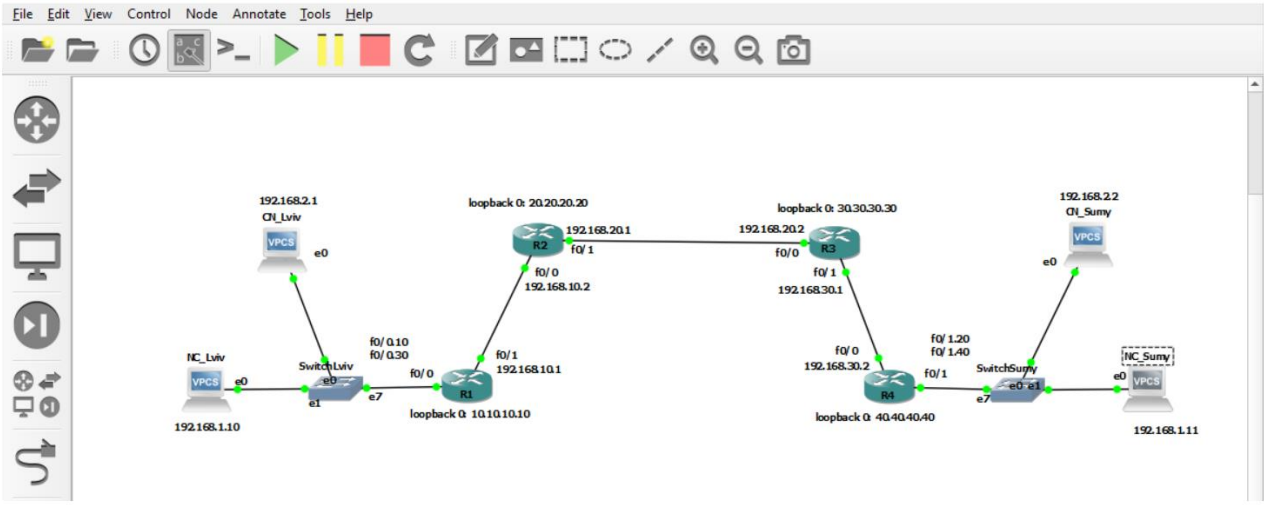

Рис 2.1. Мережа L2 VPN в GNS3

За допомогою емулятора GNS3 було побудовано мережу L2 VPN технології Carrier Ethernet. Саме для неї в рамках даної дипломної роботи буде створено графічний інтерфейс.

Для побудови L2 VPN мережі використовували наступне обладнання:

1. 2 хости VPSC
2. 4 роутери 7200 (саме вони підтримують інкапсуляцію MPLS на інтерфейсах).
3. 2 Ethernet свіча.

Налаштування свічей.

Для роботи нам потрібно налаштувати два свіча. Їх налаштування проводиться з використанням графічного інтерфейсу програми GNS3.

SW_Lviv

SW_Sumy

Налаштування роутерів.

Для побудови робочої схеми мережі необхідно вірно налаштувати роутери, вказавши інтрефейси, маршрутизацію та активацію енкапсуляції L2VPN. На

рисунках нижче розписані основні команди для кожного роутера, за допомогою яких ми будемо налаштовувати мережу.

```
en
conf t
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id loopback 0
int loopback 0
ip address 10.10.10.10 255.255.255.255
int fa 0/1
ip address 192.168.10.1 255.255.255.0
no sh
int fa 0/0
no sh
exit
int fa 0/0.10
encapsulation dot1Q 10
xconnect 40.40.40.40 9999 encapsulation mpls
int fa 0/0.30
encapsulation dot1Q 30
xconnect 40.40.40.40 8888 encapsulation mpls
router ospf 1
network 10.10.10.10 0.0.0.0 area 0
network 192.168.10.0 0.0.0.255 area 0
int fa 0/1
mpls ip
mpls mtu 1512
exit
```

Рис. 2.2. Налаштування роутера R1

```
en
conf t
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id loopback 0
int loopback 0
ip address 40.40.40.40. 255.255.255.255
int fa 0/0
ip address 192.168.30.2 255.255.255.0
no sh
int fa 0/1
no sh
exit
int fa 0/1.20
encapsulation dot1Q 20
xconnect 10.10.10.10 9999 encapsulation mpls
int fa 0/1.40
encapsulation dot1Q 40
xconnect 10.10.10.10 8888 encapsulation mpls
router ospf 1
network 40.40.40.40 0.0.0.0 area 0
network 192.168.30.0 0.0.0.255 area 0
int fa 0/0
mpls ip
mpls mtu 1512
exit
```

Рис. 2.3. Налаштування роутера R4

```
en
conf t
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id loopback 0
int loopback 0
ip address 20.20.20.20 255.255.255.255
int fa 0/0
ip address 192.168.10.2 255.255.255.0
no sh
int fa 0/1
ip address 192.168.20.1 255.255.255.0
no sh
router ospf 1
network 20.20.20.20 0.0.0.0 area 0
network 192.168.10.0 0.0.0.255 area 0
network 192.168.20.0 0.0.0.255 area 0
exit
int fa 0/0
mpls ip
mpls mtu 1512
int fa 0/1
mpls ip
mpls mtu 1512
exit
```

Рис. 2.4. Налаштування роутера R2

```
en
conf t
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id loopback 0
int loopback 0
ip address 30.30.30.30. 255.255.255.255
int fa 0/0
ip address 192.168.20.2 255.255.255.0
no sh
int fa 0/1
ip address 192.168.30.1 255.255.255.0
no sh
router ospf 1
network 30.30.30.30 0.0.0.0 area 0
network 192.168.30.0 0.0.0.255 area 0
network 192.168.20.0 0.0.0.255 area 0
exit
int fa 0/0
mpls ip
mpls mtu 1512
int fa 0/1
mpls ip
mpls mtu 1512
exit
```

Рис. 2.5. Налаштування роутера R3

Як видно з рисунків, ми виконуємо команди для збільшення mtu. Що це за параметр та навіщо збільшувати його значення?

Maximum transmission unit (MTU) це максимальний об'єм даних, які можуть бути передані протоколом за одну ітерацію. Для Ethernet ця цифра дорівнює 1500 байт (без урахування Ethernet заголовку та FCS).

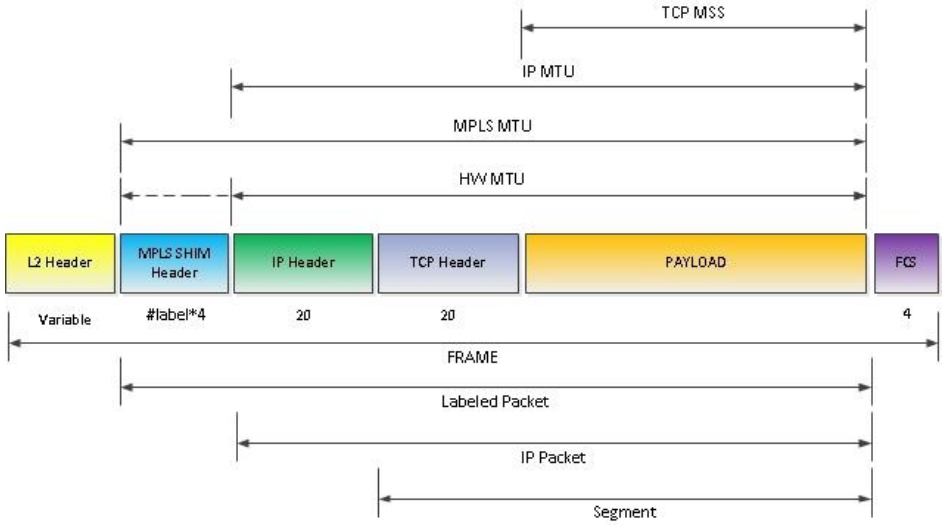

Рис. 2.6. Приклад Maximum transmission unit [20]

MPLS MTU визначає максимальний розмір поміченого IP-пакета. Якщо розмір такого пакета перевищує MPLS MTU, то такий пакет фрагментується.

MPLS MTU може бути більшим. При цьому система скоріш за все вкаже на помилку, але частіше за все вона пропустить дані. В результаті це призведе до втрати пакетів та даних [20].

Перевіряємо, що канали, які ми налаштовували, змінили стан на UP:

sh mpls 12transport vc

Налаштування хостів VPSC.

Задаємо IP-адресу та маску (Gateway задавати необов'язково)

VPCS CN_Lviv – ip 192.168.2.1 255.255.255.0

VPCS NC_Lviv – ip 192.168.1.10 255.255.255.0

VPCS CN_Sumy – ip 192.168.2.2 255.255.255.0

VPCS NC_Sumy – ip 192.168.1.11 255.255.255.0

2.2 Wireshark – програма для аналізу мережевого трафіку

Найбільш розповсюджена програма для захоплення та аналізу мережевого трафіка. Вона набула своєї популярності завдяки тому що працює практично зі всіма мережевими протоколами, а також надає користувачу максимально можливу інформацію про перехоплений трафік. За її допомогою досліджуються протоколи та пакети. Крім того вона допомогає нам зрозуміти можливі помилки при побудові мережі та виправити їх.

Одна із особливостей емулятора GNS3 – це інтегрований в неї Wireshark. Це не лише зручно, це дозволяє зекономити багато часу, що був би витрачений на інсталіцяю та вивчення інтрфейсу програми.

Програма має дуже вдалий графічний інтерфейс, завдяки якому досить легко проаналізувати отримані дані. Не вимагає ніякого додаткового обладнання та особливих налаштувань.

Ще однією позитивною відмінністю Wireshark є, створена розробниками, можливість працювати з трафіком в офф-лайн режимі. Це дуже зручно, оскільки відсутня постійна прив’язка до інтренету. Дані можна аналізувати в будь-який час та повертатися до них неодноразово.

Робота з Wireshark в межах GNS3 дуже проста і зрозуміла і допомогла нам побачити як працює інкапсуляція в технології L2 VPN MPLS. Ми побачили як налаштовуються мітки та який їх шлях через маршрутизатори мережі при даній технології.

В отриманому перехопленому трафіку ми аналізували ICMP-пакети (replay і request) і чітко змогли побачити зовнішні та внутрішні мітки і різницію між ними.

На малюнках, що наведені нижче, чітко видно подвійну мітку ICMP-пакету. Верхня мітка – це зовнішня мітка або мітка маршрутизації. Нижня мітка – це внутрішня мітка або мітка організації. Як видно існує різниця в мітках в залежності від призначення пакета.

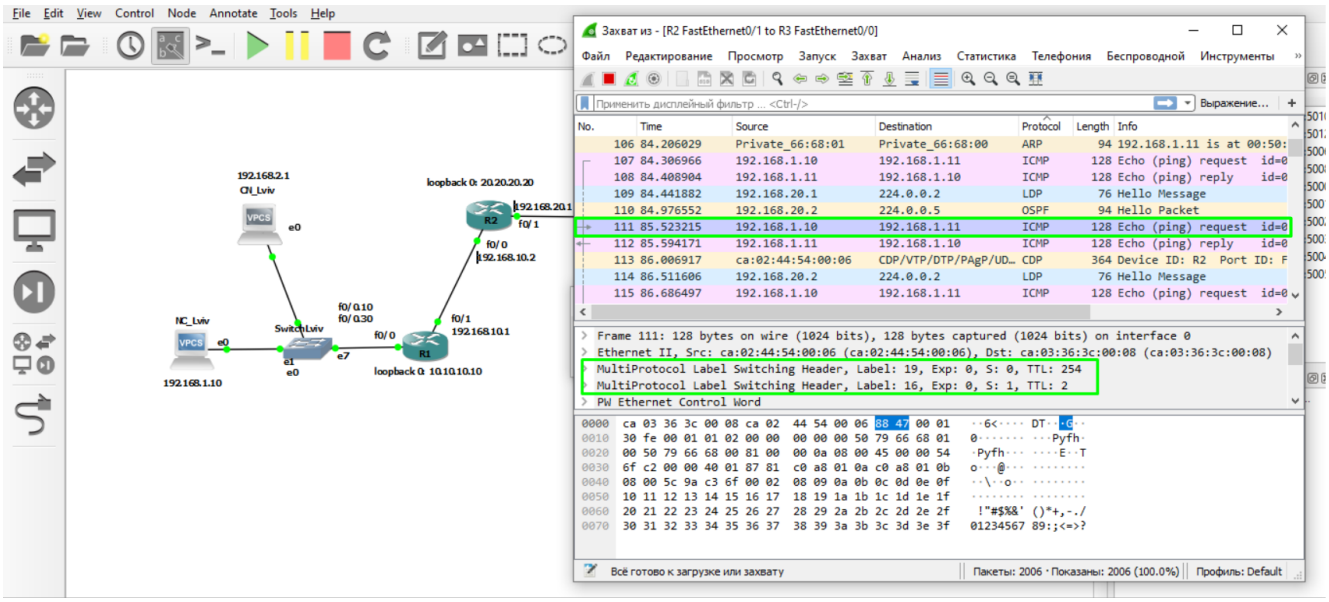

Рис. 2.7. Перехоплений Echo (ping) request пакет Wireshark (R2-R3)

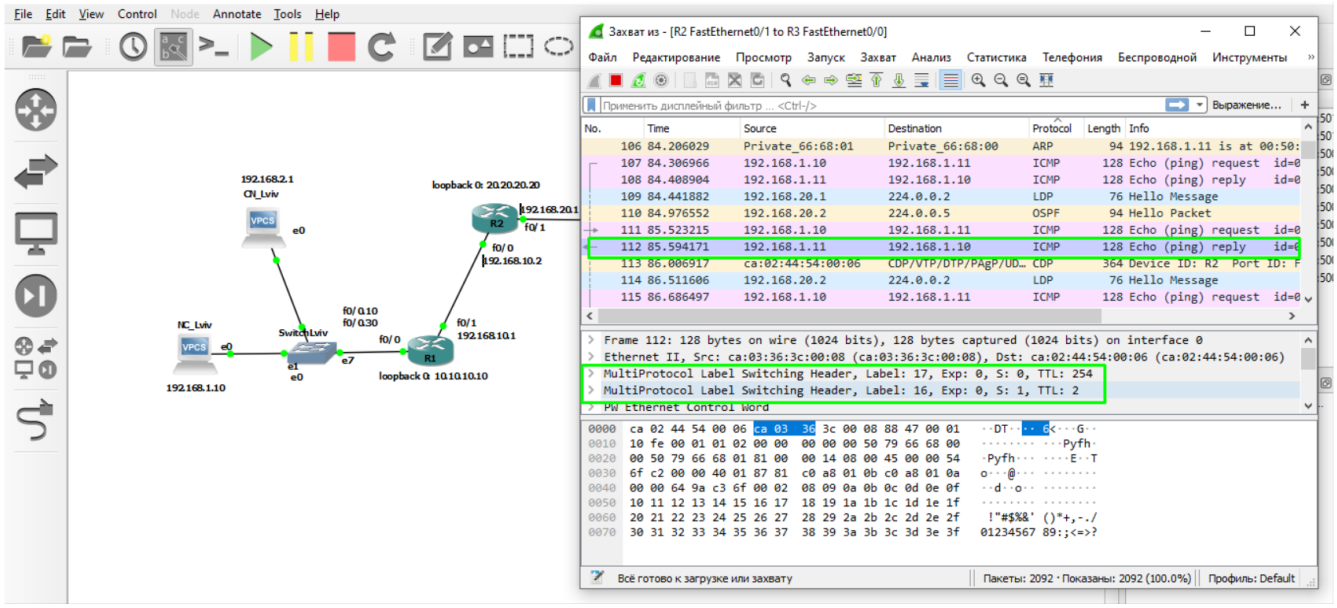

Рис 2.8. Перехоплений Echo (ping) reply пакет Wireshark (R2-R3)

Крім того, ми побачили що існує різниця в мітках пакету в залежності від того, де ми його перехопили. На прикладах, наведених вище, ми перехопили пакет між роутерами R2 та R2. Побачили наявність двох міток в обох пакетах.

На рисунках, що наведені нижче, ми перехопили пакет між роутерами R1та R2.

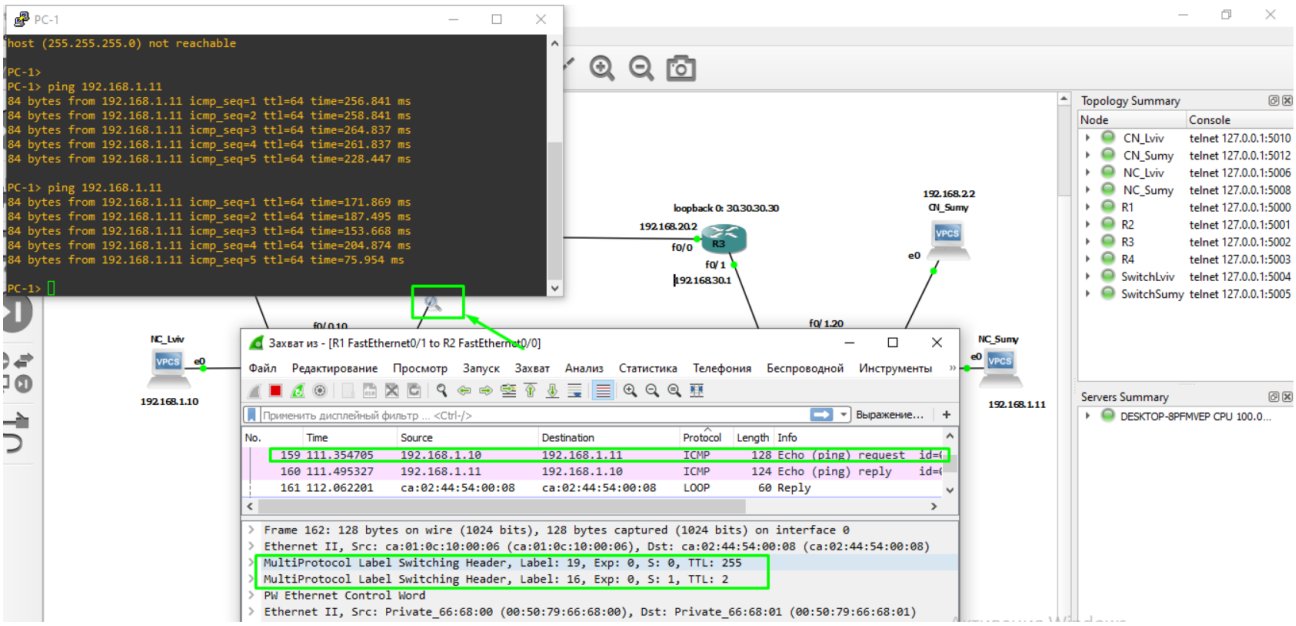

Рис. 2.8. Перехоплений Echo (ping) request пакет Wireshark (R1-R2)

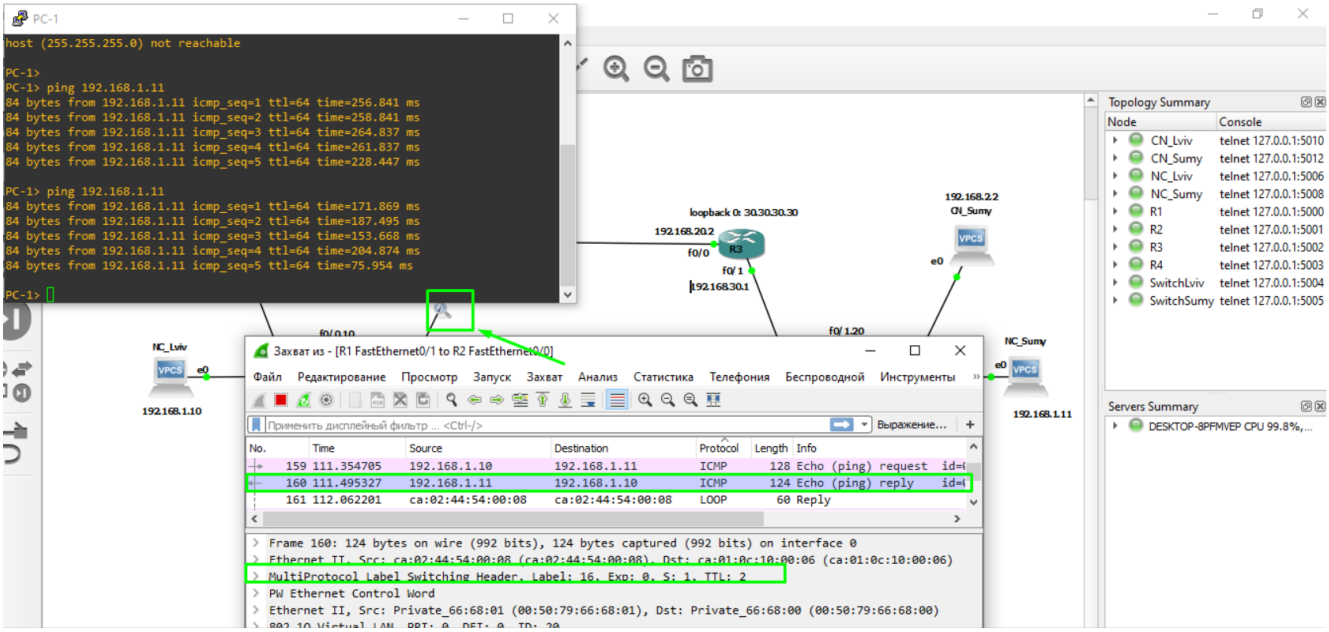

Рис. 2.9. Перехоплений Echo (ping) reply пакет Wireshark (R1-R2)

ICMP-пакет, що ми відправляли з NC_Lviv до NC_Sumy не містіть ніяких нових відмінностей, порівняно з поперденім прикладом. А ось зворотній пакет Echo (ping) reply вже має відмінність, що полягає в наявності лише однієї мітки. Це ще

раз підтверджує теорію про те, що граничному роутеру передається вже пакет без додаткових міток.

Проаналізувавши роботу мережі за допомого Whireshark, стає очевидним що мережа налаштована вірно, інкапсуляція в середині мережі працює згідно правилам і технологія L2 VPN MPLS реалізована повною мірою.

2.3 Побудова графічного інтерфейсу за допомогою мови JAVAScript

Головною метою дипломної роботи була побудова графічного інтерфейсу для налаштувать маршрутизаторів при створенні мережі Carrier Ethernet. Найбільш типовим для такої роботи виявився JavaScript.

JavaScript (JS) – це об’єктно-орієнтована скриптована мова програмування, що використовується у веб-розробках. Функціональність JS вражає і дає можливість реалізувати велику кількість сценарієв.

У веб браузері JS виконує все, що пов’язано з маніпулюванням веб-сторінок при взаємодії користувача з веб-сервером. І має широкий спектр можливостей. Але є і обмеження роботи JS в браузері, що обумовлено безпекою та конфеденційностю даних користувачів.

JavaScript вважається найпоширенішим інструментом для створення браузерних інтрефейсів завдяки тому, що має повну інтеграцію з HTML/CSS та підтримується усіма популярними браузерами. До того ж з JS досить легко працювати, тому опанувати її можуть навіть школярі.

Серед основних особливостей JavaScript, які саме і дозволили реалізувати розробку графічного інтерфейсу, слід відмітити:

- можливість зберігання даних всередині змінних;
- операції з текстовими даними;
- можливість запускати код як результат певних дій на веб сторніці.

Скрипти JavaScript виконуються на веб-сторінці інтерпретатором браузера. JS вважається клієнт-орієнтовною технологією, оскільки всі ці операції відбуваються на стороні клієнта.

За допомогою JavaScript вдалося побудувати зручний графічний інтерфейс для налаштування мережі. Користувач зможе вводити потрібні йому дані (в нашому випадку це будуть IP-адреси та маски), та отримувати налаштування маршрутизаторів для цих данних. Саме вони можуть бути потім використані для налаштувань реального обладнання або віртуальних маршрутизаторів в рамках різних мережевих емуляторів.

РОЗДІЛ 3
СТВОРЕННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ ДЛЯ НАЛАШТУВАННЯ L2 VPN МЕРЕЖІ

3.1 Розробка графічного інтерфейсу налаштування технології Carrier Ethernet

В другому розділі ми побудували необхідну нам для практичної часпини схему L2 VPN мережі

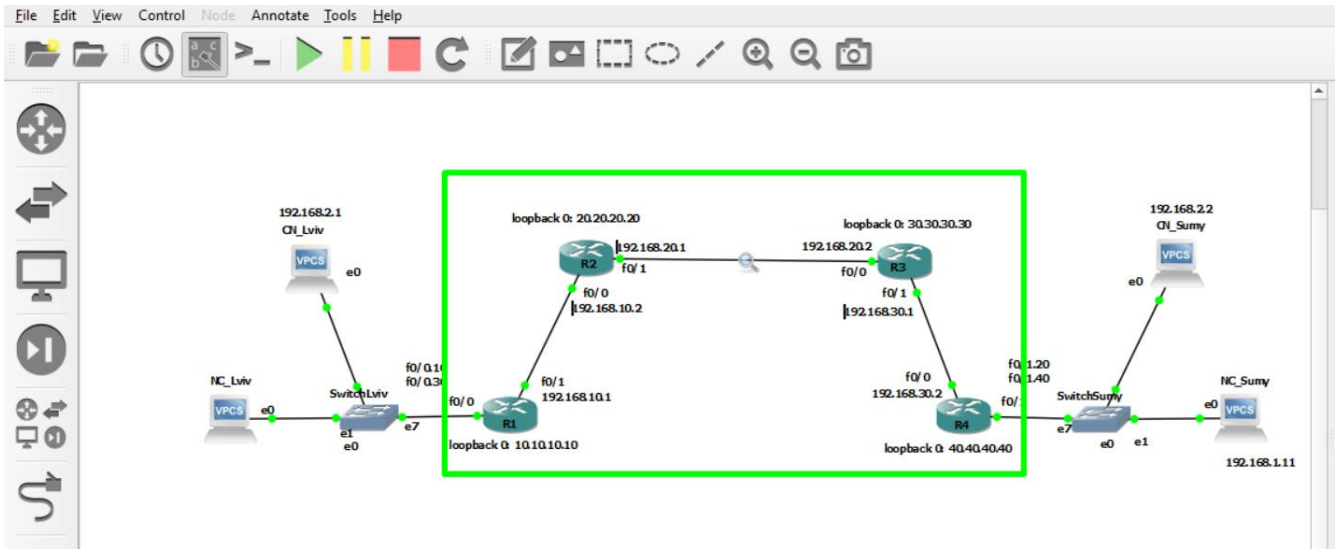

Рис. 3.1. Налаштування L2VPN мережі

Головна проблема при створенні такої мережі полягає в складності налаштування роутерів. Саме вони потребують глибокого розуміння процесів, знання команд та досвід в роботі з подібними схемами. Як можна було побачити у попередньому розділі, команди для налаштування роутерів досить складні і різні роутери налаштовуються по-різному. Крім того, потрібно рузміти логіку роботи обладнання та його з’єднання.

Зі схеми на малюнку зрозуміло, що роутери R1 та R4 схожі і налаштовуються однаково. Теж саме стосується і роутерів R2 та R3.

Виходячи з цього, графічний інтефейс буде розроблений для роутреів R1 та R2. Налаштування для роутерів R3 та R4 можна бути легко отримати якщо ввести саме їх параметри в поля графічного інтрефейсу.

Проект, що було створено для реалізації головної мети даної диплоної роботи можна знайти в додатках.

Отриманий графічний інтерфейс простий та зрозумілий. Схема Ethernet мережі досить умовна, для того щоб не лякати користувача та неперевантажувати його додатковими елементами.

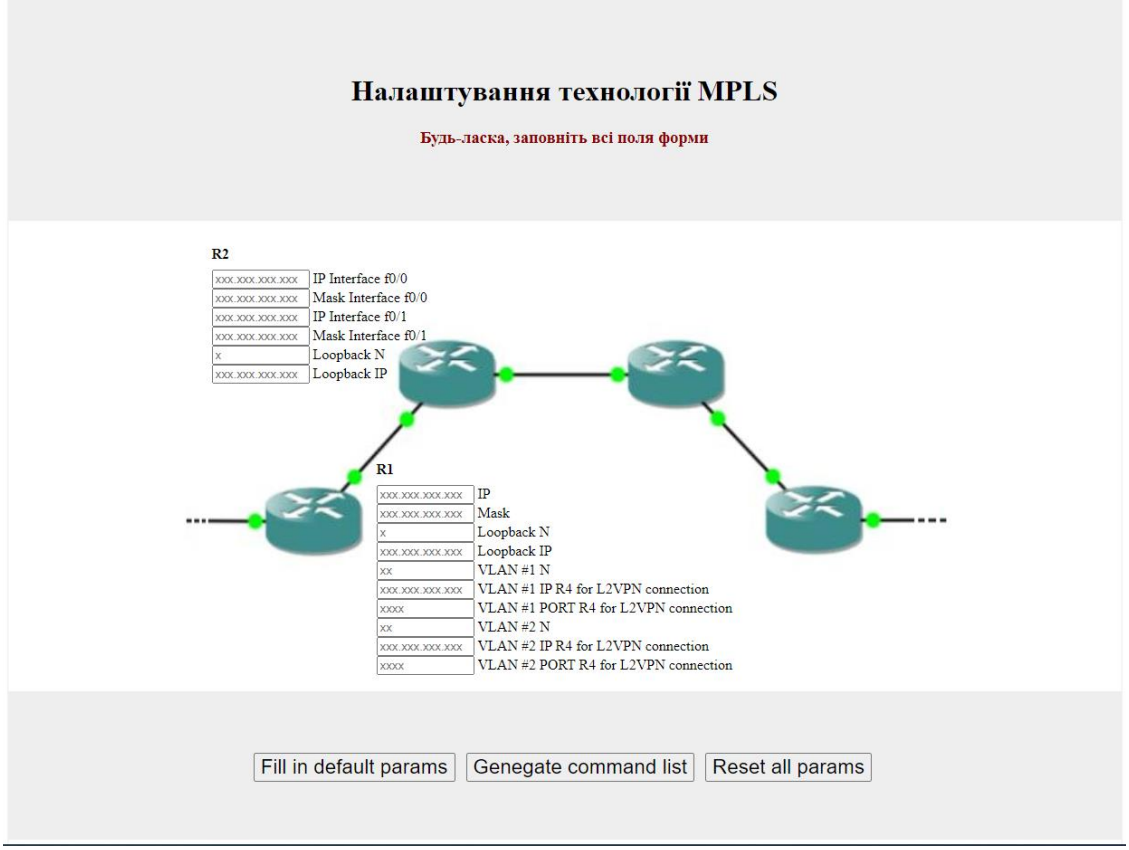

Рис. 3.2. Графічний інтерфейс налаштування технології MPLS

Як видно з рисунка 3.2, графічний інтерфейс представляє собою набор полів для введення змінних параметрів мережі та кнопок для запуску програми.

Всі поля, куди користувач повинен ввести дані, мають зрозумілу назву і певну послідовність. Це допоже користувачу швидше зорієнтуватися в тому, що і куди треба вводити.

Поля для IP-адрес та масок мережі мають валідаційну перевірку. Якщо буде введений невірний формат даних, користувач побачить повідомлення про це і виконати процедуру не вдасться.

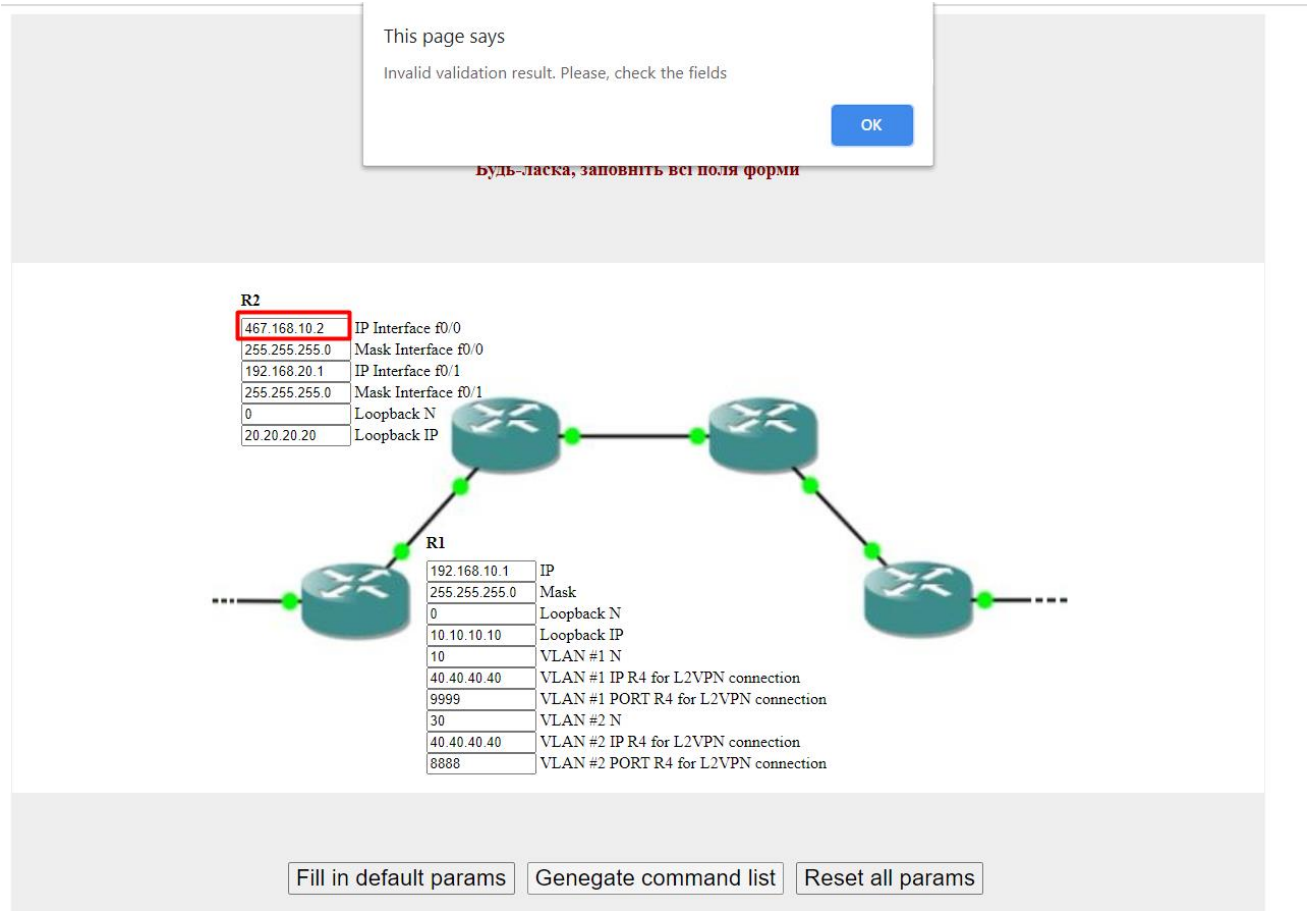

Рис. 3.3. Валідація на невірно введену IP-адресу

Кнопки для роботи з графічним інтрефейсом настпуні:

– Fill in default params. Кнопка для заповнення полів значеннями за замовчуванням. Додадткова опція для тих, хто не розуміє що і як заповнювати або не хоче розбиратися яку IP-адресу можна використати для налаштувань;

– Generate command list. За допомогою цієї кнопки саме і відбувається генерація списку команд, який потім можна буде використовувати у налаштуванні роутерів;
– Reset all params. Ця кнопка очищує всі поля від введних користувачем даних.

Після того як користувач вірно заповнив всі дані і нажав на кнопку [Generate command list], він отримує налаштування для кожного роутера окремо, як показано на рисунку 3.4

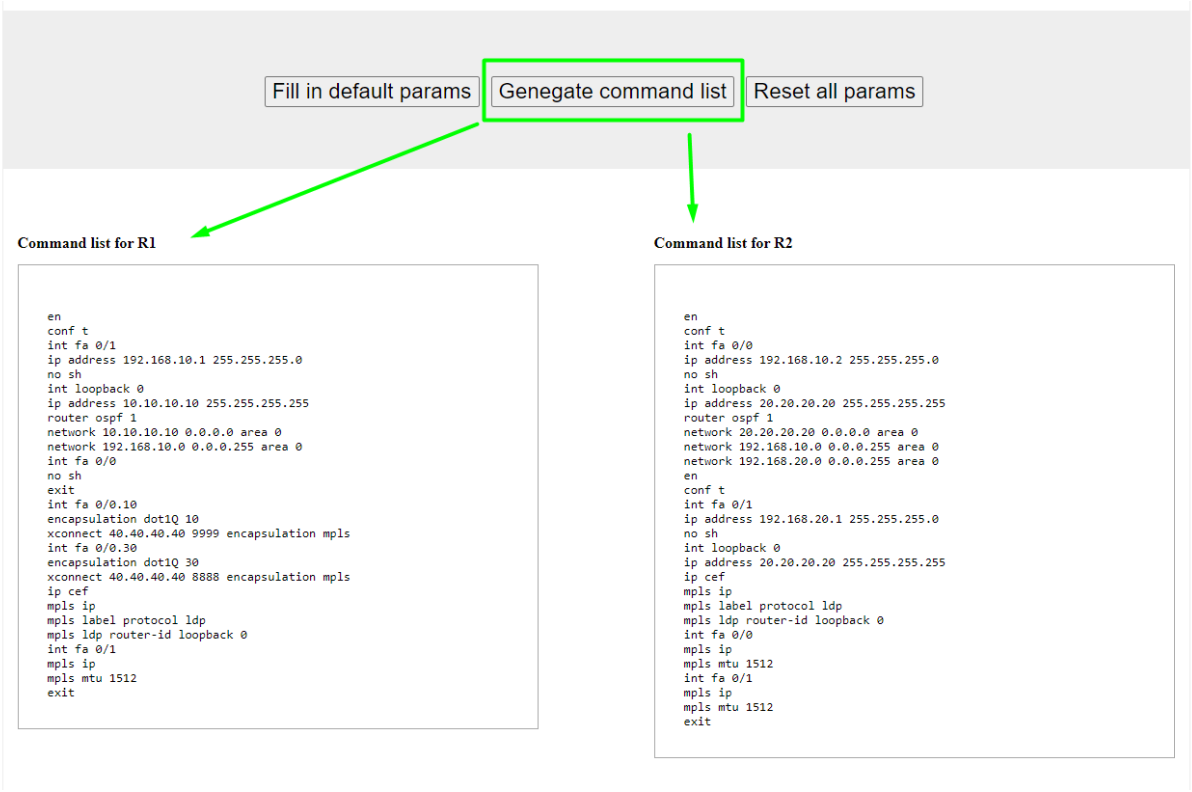

Рис. 3.4. Виведення команд налаштування роутерів

Тепер можна легко скопіювати ці налаштування і ввести їх на реальне обладнання або в емуляторі і отримати готовий налаштований роутер або зберегти в окремий файл.

3.2 Тестування графічного інтерфейсу Carrier Ethernet

Після того як ми отримали дані для налаштування роутерів, ми можемо їх використати для налаштування наших роутерів R1 та R2 в емуляторі GNS3.

Для цього заходимо по черзі на кожний роутер в режим консолі, вставляємо, отриманий через графічний інтерфейс, набір налаштувань і застосовуємо.

Якщо всі поля в графічному інтрефейсі були заповнені вірно, консоль прийме всі команди на виконання без зауважень. Наступні малюнки ілюструють процес введення команд в консолі роутера.

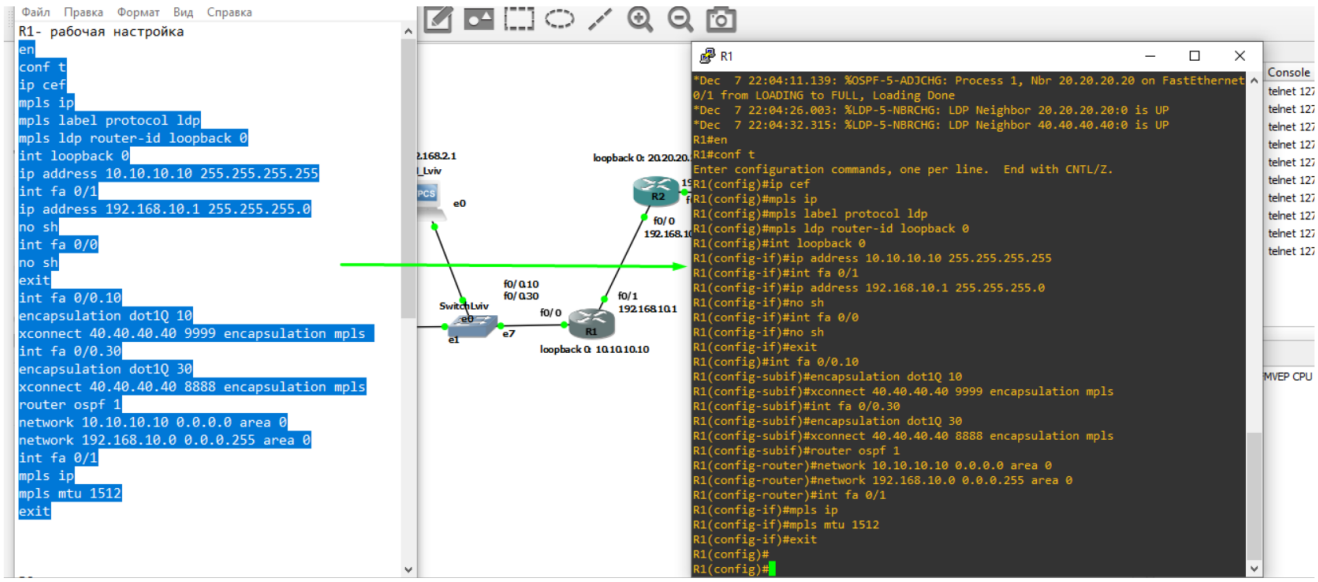

Рис. 3.5. Налаштування роутера R1

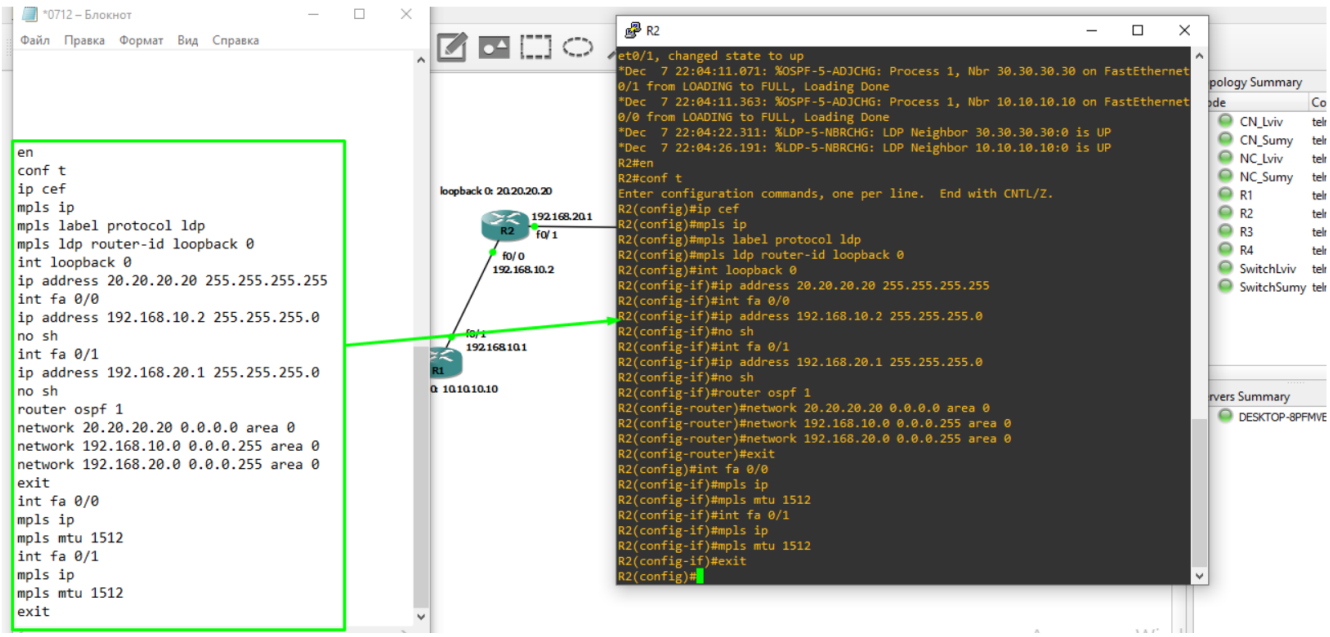

Рис. 3.6. Налаштування роутера R2

Після того як ми застосували ці налаштування на роутерах у емуляторі, нам потрібно перевірити їх правильність. Це ми легко зможемо зробити за допомогою звичайної команди пінгування комп'ютерів.

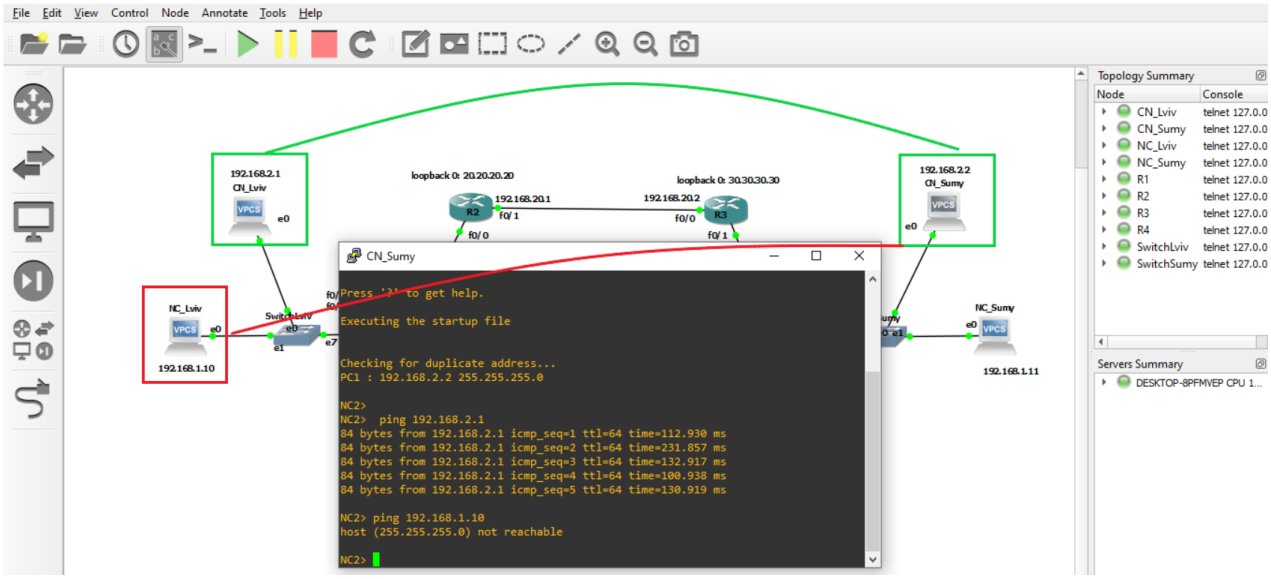

Рис. 3.7. Перевірка доступності мережі для компанії CN

Як видно із консолі, ком'ютери компанії CN пінгуються як і повинні, а хости компанії NC недоступні. Тепер перевіримо як працює мережа для компанії NC. Для цього відправимо пакети з NC_Lviv до NV_Sumy та CN_Sumy. З консолі на малюнку 3.8 бачимо, що мережа працює передбачувано. Пакети NC_Lviv до NV_Sumy були доставлені, а хост компанії CN_Sumy є недосяжним.

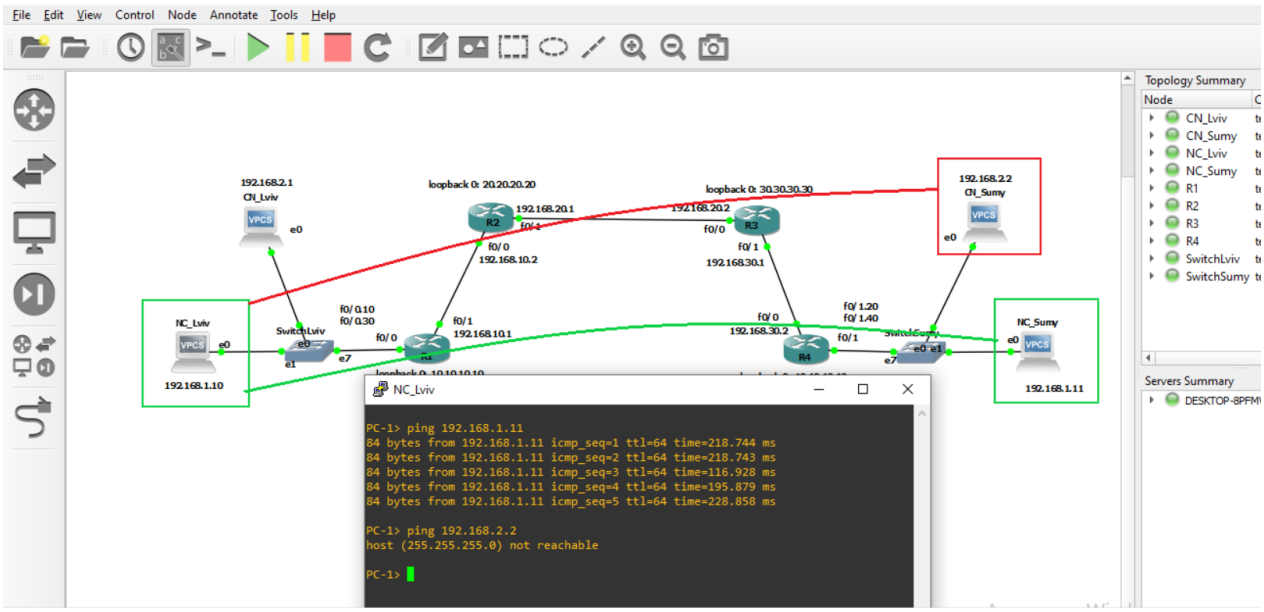

Рис. 3.8. Перевірка доступності мережі для компанії CN

Тепер ми бачимо, що якщо порівняти налаштування мережі за допомогою команд, що представлені у другому розділі і команд, які отримали за допомогою графічного інтрефейсу, різниця очевидна.

При налаштуванні роутерів звичайним способом, без допомоги інтерфейсу, доведеться розібратися з декількома великими главами хрестоматійних книжок по принципам налаштування мереж. І глибоко розібратися з тим як працює технологія Carrier Ethernet.

Якщо ж намагатися зробити те саме через графічний інтерфейс, достатньо зрозуміти як ним користуватися та як ввести в емуляторі отпримані і скопійовані налаштування.

ВИСНОВОК

Виходячи з мети, яка була сформована на почтаку роботи, можна зробити наступні висновки:

- Були розглянуті особливості роботи емулятора GNS3. Стандартні налаштування допомогли побудувати мережу за технологією Carrier Ethernet;
- За допомогою програми Wireshark були проаналізовані ICMP-пакети. З цього анілізу став зрозумілим принцип додавання та видалення пакетів при транспорті їх між магістральними роутерами. Детальний аналіз пакету, що був перехоплений програмою, показав наявність у кадрі двох міток – клієнтської та операторської;
- В рамках роботи був розроблений графічний інтерфейс, який значно полегшить налаштування мережі технології Carrier Ethernet. Результатом роботи графічного інтерфейсу є налаштування для роутерів, які можуть бути застосовані навіть користувачами без досвіду та глибоких знань інформаційно-телекомунікаційних технологій. Інтерфейс може бути використаний як для роботи в емуляторах, так і на реальному обладнанні;
- Даний інтерфейс може використовуватися в навчальному процесі в школах, університетах та спеціалізованих гуртках. Також він стане в нагоді адміністраторам-початківцям без достатнього досвіду в налаштуванні такого типу мереж.

СПИСОК ЛІТЕРАТУРИ

1. Олифер Н. Компьютерные сети. Принципы, технологии, протоколы, / В. Олифер, Н. Олифер. – Спб.: Питер, 2020. – 1008 с. – (Юбилейное издание).

2. Эффективный механизм передачи данных в опорных IP-сетях с использованием технологии MPLS // Беспроводные технологии. – 2017. – № 4. – С. 14-20.

3. Обґрунтування впровадження технології мультипротокольної комутації по міткам як основи транспортної мережі зв'язку // Сучасні інформаційні технології у сфері безпеки та оборони. – 2014. – № 3. – С. 64-68.

4. MPLS – как работает и зачем нужен [Електронний ресурс] - https://wiki.merionet.ru/seti/25/mpls-kak-rabotaet-i-zachem-nuzhen/

5. Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field [Електронний ресурс] - https://tools.ietf.org/html/rfc54622

6. MPLS сделает маршрутизаторы быстрыми [Електронний ресурс] - https://compress.ru/article.aspx?id=10621

7. Вивек Ольвейн. Структура и реализация MPLS / Вивек Ольвейн: [пер. с англ.] – М. : изд. дом «Вильямс», 2004. – 480 с.

8. MPLS L2VPN Configuration [Електронний ресурс] - Support - 04-MPLS L2VPN Configuration- H3C

9. Сети VPN и технология MPLS [Електронний ресурс] - https://www.lessons-tva.info/archive/nov030.html

10. Сети для самых матёрых. Часть двенадцатая. MPLS L2VPN [Електронний ресурс] - https://habr.com/ru/post/315028/

11. Базовые сервисы технологии MPLS [Електронний ресурс] - https://nag.ru/articles/reviews/15448/bazovyie-servisyi-tehnologii-mpls.html

12. Сети для самых маленьких. Часть десятая. Базовый MPLS [Електронний ресурс] - https://habr.com/ru/post/246425/

13. Словарь телекоммуникационных терминов [Електронний ресурс] - http://lookmeup.linkmeup.ru/#term487

14. Что такое Double VLAN (Q-in-Q) и примеры настройки [Електронний ресурс] - https://www.dlink.ru/ru/faq/62/237.html

15. 802.1Q Tunneling (Q-in-Q) Configuration [Електронний ресурс] - https://networklessons.com/switching/802-1q-tunneling-q-q-configuration-example

16. Лекция 11:Передача данных с коммутацией по меткам [Електронний ресурс] - https://intuit.ru/studies/courses/1123/200/lecture/5195?page=10

17. Ethetnet поверх Ethetnet [Електронний ресурс] - http://iptcp.net/ethernet-poverkh-ethernet.html

18. Основы GNS3. Обзор [Електронний ресурс] - https://habr.com/ru/post/266503/

19. Новый GNS3 – шаг вперед или прыжок на месте? [Електронний ресурс] - https://habr.com/ru/post/244955/

20. Maximum Transmission Unit (MTU). Мифы и рифы [Електронний ресурс] - https://habr.com/ru/post/226807

ДОДАТОК

```
$(document).ready(function(){
   var r1_command_list = `
en
conf t
int fa 0/1
ip address {r1_ip} {r1_mask}
no sh
int loopback {r1_loopback_number}
ip address {r1_loopback_ip} 255.255.255.255
router ospf 1
network {r1_loopback_ip} 0.0.0.0 area 0
network {r1_net_ip} 0.0.0.255 area 0
int fa 0/0
no sh
exit
int fa 0/0.{r1_vlan1_number}
encapsulation dot1Q {r1_vlan1_number}
xconnect {r1_vlan1_connection_ip} {r1_vlan1_connection_port}
encapsulation mpls
int fa 0/0.{r1_vlan2_number}
encapsulation dot1Q {r1_vlan2_number}
xconnect {r1_vlan2_connection_ip} {r1_vlan2_connection_port}
encapsulation mpls
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id loopback {r1_loopback_number}
int fa 0/1
mpls ip
mpls mtu 1512
exit`;
   var r2_command_list = `
en
conf t
int fa 0/0
ip address {r2_ip_1} {r2_mask_1}
no sh
int loopback {r2_loopback_number}
ip address {r2_loopback_ip} 255.255.255.255
int fa 0/1
```

```
ip address {r2_ip_2} {r2_mask_2}
no sh
int loopback {r2_loopback_number}
ip address {r2_loopback_ip} 255.255.255.255
router ospf 1
network {r2_loopback_ip} 0.0.0.0 area 0
network {r2_net1_ip} 0.0.0.255 area 0
network {r2_net2_ip} 0.0.0.255 area 0
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id loopback {r2_loopback_number}
int fa 0/0
mpls ip
mpls mtu 1512
int fa 0/1
mpls ip
mpls mtu 1512
exit`;
   var field_map = {
      'r1_ip': {label:'IP', v:'192.168.10.1', t:'ip'},
      'r1_mask': {label:'Mask', v:'255.255.255.0', t:'ip'},
      'r1_loopback_number': {label:'Loopback N', v:'0', t:'int'},
      'r1_loopback_ip': {label:'Loopback IP', v:'10.10.10.10', t:'ip'},
      'r1_vlan1_number': {label:'VLAN #1 N', v:'10', t:'int'},
      'r1_vlan1_connection_ip': {label:'VLAN #1 IP R4 for L2VPN
connection', v:'40.40.40.40', t:'ip'},
      'r1_vlan1_connection_port': {label:'VLAN #1 PORT R4 for L2VPN
connection', v:'9999', t:'int'},
      'r1_vlan2_number': {label:'VLAN #2 N', v:'30', t:'int'},
      'r1_vlan2_connection_ip': {label:'VLAN #2 IP R4 for L2VPN
connection', v:'40.40.40.40', t:'ip'},
      'r1_vlan2_connection_port': {label:'VLAN #2 PORT R4 for L2VPN
connection', v:'8888', t:'int'},
      'r2_ip_1': {label:'IP Interface f0/0', v:'192.168.10.2', t:'ip'},
      'r2_mask_1': {label:'Mask Interface f0/0', v:'255.255.255.0', t:'ip'},
      'r2_ip_2': {label:'IP Interface f0/1', v:'192.168.20.1', t:'ip'},
      'r2_mask_2': {label:'Mask Interface f0/1', v:'255.255.255.0', t:'ip'},
      'r2_loopback_number': {label:'Loopback N', v:'0', t:'int'},
      'r2_loopback_ip': {label:'Loopback IP', v:'20.20.20.20', t:'ip'}
```

```
        };

        Object.keys(field_map).forEach(function(name) {
          document.getElementById(name).placeholder = (field_map[name].t
== 'ip')
            ? 'xxx.xxx.xxx.xxx' : field_map[name].v.replaceAll(/./gi, 'x');
          var newlabel = document.createElement("Label");
          newlabel.setAttribute("for", name);
          newlabel.innerHTML = field_map[name].label;

document.getElementById(name).parentElement.appendChild(newlabel);
        });

        $('#set_default').click(function(){
          // console.log('set_default click');
          Object.keys(field_map).forEach(function(name) {
            document.getElementById(name).value = field_map[name].v;
          });
          return;
        });
        $('#reset_all').click(function(){
          // console.log('reset_all click');
          Object.keys(field_map).forEach(function(name) {
            document.getElementById(name).value = '';
          });
          return;
        });
        $('#gen_commands').click(function(){
          // console.log('gen_commands click');
          if (validate_fields(field_map)) {
            document.getElementById('r1_output').innerHTML =
replace_marks(r1_command_list);
            document.getElementById('r2_output').innerHTML =
replace_marks(r2_command_list);
          } else {
            alert('Invalid validation result. Please, check the fields');
          }
          return;
        });
      });
```

```
function validate_fields(field_map) {
    var is_validate = true;
    Object.keys(field_map).forEach(function(name) {
        var el = document.getElementById(name);
        if (!el || !el.value) {
            // console.log('mandatory err')
            is_validate = false;
            return;
        }
        validator = field_map[name].t;
        if (validator == 'ip' && !/^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/.test(el.value)) {
            // console.log('IP err');
            is_validate = false;
        }
        if (validator == 'int' && !/^\d+$/.test(el.value)) {
            // console.log('IP err');
            is_validate = false;
        }
    });
    return is_validate;
}

function replace_marks(command_list) {
    var mapping = {
        'r1_net_ip': ['r1_ip', 'r1_mask'],
        'r2_net1_ip': ['r2_ip_1', 'r2_mask_1'],
        'r2_net2_ip': ['r2_ip_2', 'r2_mask_2']
    };
    command_list.match(/{([^}]+)}/gi).forEach(function(mark) {
        var html_id = mark.slice(1,-1);
        var v = Object.keys(mapping).includes(html_id)
            ? get_net_ip(
                document.getElementById(mapping[html_id][0]).value,
                document.getElementById(mapping[html_id][1]).value
            )
            : document.getElementById(html_id).value;
        command_list = command_list.replaceAll(mark, v);
```

```
        })
        return command_list;
    }

    function get_net_ip(ip, mask) {
        var ip_as_str = ip_to_int(ip).toString(2).padStart(32, '0');
        var int_mask = (~ ip_to_int(mask)).toString(2).length;
        var net_ip_as_str = ip_as_str.substring(0, 32 - int_mask) +
'0'.repeat(int_mask);
        return int_to_ip(parseInt(net_ip_as_str, 2));
    }

    function ip_to_int(ip) {
        return ip.split('.').map((octet, index, array) => {
          return parseInt(octet) * Math.pow(256, (array.length - index - 1));
        }).reduce((prev, curr) => {
          return prev + curr;
        });
    }

    function int_to_ip(value) {
        return [
          (value >> 24) & 0xff,
          (value >> 16) & 0xff,
          (value >> 8) & 0xff,
          value & 0xff
        ].join('.');
    }
```