УДК 621.39, 004.7
КП
№ держреєстрації 0108U008900
Інв. №

Міністерство транспорту та зв'язку України
Державна адміністрація зв'язку
Одеська національна академія зв'язку ім. О.С. Попова
(ОНАЗ ім. О.С. Попова)
65029, м. Одеса, вул. Ковальська, 1; тел. (048) 731-75-35

ЗАТВЕРДЖУЮ
Ректор, д.т.н, професор
_____ П.П. Воробієнко
2009.11.16

ЗВІТ
ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ
РОЗРОБКА НОВИХ СИСТЕМ АДРЕСАЦІЇ ГЛОБАЛЬНИХ МЕРЕЖ

(заключний)

Начальник НДЧ
к.е.н. _____ І.В. Яцкевич
2009.11.16

Керівник НДР
д.т.н., професор, _____ Д.А. Зайцев
зав. кафедри ПЗМЗ 2009.11.13

2009

СПИСОК АВТОРІВ

Керівник НДР д.т.н., професор, зав. каф. ПЗМЗ	Д.А. Зайцев (вступ, розділи 1-5, висновки, рекомендації)
Відповідальний виконавець к.т.н., доцент каф. КС	Т.Р. Шмельова (реферат, розділи 2-4, перелік посилань)
Інженер	К.Д. Гуляєв (розділи 1-5, додатки)

РЕФЕРАТ

Звіт про НДР: 124 с., 51 рис., 18 табл., 73 джерел – складається з вступу, 5 розділів, висновків та рекомендацій, 2 додатків.

Об'єктом дослідження є процеси функціонування мереж з комутацією пакетів; предметом дослідження є система адресації глобальних мереж, стек мережних протоколів, алгоритми доставки пакету, архітектура та програмне забезпечення мережних пристроїв. Метою роботи є створення уніфікованої системи адресації глобальних мереж з пакетною комутацією, яка забезпечує гарантовану якість обслуговування, підвищує корисну продуктивність, розширює адресний простір. Запропоновано нову систему адресації глобальних мереж Е6 та її модифікації; розроблено відповідний стек протоколів та специфікації використаних пристроїв і програмного забезпечення. Єдина мережна адреса Е6 довжиною 6 октетів та ієрархічною структурою використається замість IP адрес і MAC адрес; скасовано протоколи TCP та IP; для гарантованої доставки інформації використане засоби Ethernet LLC2. Отримано оцінки підвищення корисної продуктивності в 1,6 разів та скорочення часу доставки пакету в 4 рази; простір мережних адрес розширено в 16 тис. разів стосовно IP. Розроблено моделі Е6 мереж з динамічно-векторною маршрутизацією та тимчасовим вимкненням пристроїв у вигляді розфарбованих сітей Петрі. Виконано моделювання PBB мереж та порівняльний аналіз отриманих результатів стосовно Е6 адресації. Знайдено, що суттєвим недоліком технології PBB є непередбачені тимчасові перевантаження мережі зумовлені широкомовними штормами, що унеможливлює гарантовану якість обслуговування. Розроблено стратегію практичного впровадження Е6 мереж та організації шлюзів з TCP/IP мережами. Виконано експериментальну програмну реалізацію стеку протоколів Е6 в середовищі ядра операційної системи Linux та спостерігання передачі Е6 пакетів в мережі за допомогою аналізаторів трафіку. Результати роботи призначені для застосування в якості нової технології глобальних мереж. Пропонується продовжити дослідження та виконати промислову реалізацію Е6 мереж, що виведе Україну в лідери інформаційно-комунікаційних технологій.

ЄДИНА МЕРЕЖНА АДРЕСА Е6, ДИНАМІЧНА МАРШРУТИЗАЦІЯ, ДИСТАНЦІЙНО-ВЕКТОРНІ АЛГОРИТМИ, МАГІСТРАЛЬНІ МОСТИ ПРОВАЙДЕРА PBB, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, РОЗФАРБОВАНІ СІТІ ПЕТРІ, ГАРАНТОВАНА ЯКІСТЬ ОБСЛУГОВУВАННЯ, ОПЕРАЦІЙНА СИСТЕМА LINUX, СТЕК ПРОТОКОЛІВ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Позначення	Розшифровка	Опис
CIDR	Classless Inter-Domain Routing	Безкласова маршрутизація всередині доменів
DHCP	Dynamic Host Configuration Protocol	Протокол динамічної конфігурації хоста
DNS	Domain Name System	Система імен доменів
DWDM	Dense Wave Division Multiplexing	Мультіплексування з плотним розподіленням довжини хвиль
E6	Uniform Network Address	Єдина мережна адреса
FTP	File Transfer Protocol	Протокол передачі файлів
IP	Internet Protocol	Протокол Інтернету
HTTP	Hypertext Transfer Protocol	Протокол передачі гіпертексту
MPLS	Multiprotocol Label Switching	Багато протокольна комутація позначок
NAT	Network Address Translator	Транслятор адрес мережі
PBB	Provider Backbone Bridge	Магістральний міст провайдера
QoS	Quality of Service	Якість обслуговування
RIP	Routing Information Protocol	Протокол маршрутної інформації
SMTP	Simple Mail Transfer Protocol	Простий протокол передачі електронної пошти
TCP	Transmission Control Protocol	Протокол керування передачі даних
UDP	User Datagram Protocol	Протокол датаграм користувача
URL	Uniform Resource Locator	Уніфікований локатор ресурсу
VoIP	Voice over IP	Передача голосу через Інтернет протокол

ВСТУП

На теперішній час склалися певні проблеми подальшого розвитку глобальних мереж зумовлені системою адресації та алгоритмами доставки пакетів, для розв'язання яких було запропоновано IPv6 для розширення адресного простору, MPLS та PBB для прискореної доставки пакетів, численні методи інжинірингу трафіку для забезпечення якості обслуговування.

Серед основних тенденцій розвитку сучасних мереж слід зазначити наступні:

- переважне застосування користувачем безадресного доступу та імен ресурсів (URL, DNS);
- домінування на прикладному рівні застосувань стеку протоколів TCP/IP в Інтернет/Інтранет;
- розвиток додаткових технологій прискореної доставки пакетів в магістралях (MPLS, PBB);
- домінування на канальному (фізичному) рівні технології Ethernet та її розширень.

Але технологія інкапсуляції IP пакетів в кадри Ethernet залишається в основному такою, як це було передбачено в роботах Джона Постела, Девіда Плюмера та Чарза Хорніга в 1980-1984 роках. Нові технології такі як MPLS та PBB лише доповнюють попередні стандарти. Відображення адрес історично грало позитивну роль для інтеграції різних канальних технологій в гетерогенних мережах, але за теперішніх часів в умовах домінування на канальному рівні технології Ethernet є в більшості випадків надлишковим. Основною перешкодою на шляху забезпечення гарантованої якості обслуговування є відображення адрес втілене в протоколах ARP/RARP, які можуть вносити суттєву затримку до часу доставки пакету, а також широкомовлення та алгоритми покривного дерева застосовані для доставки кадру Ethernet.

Основними проблемами розвитку сучасних мереж є:

- забезпечення гарантованої якості обслуговування;
- бракування адрес;
- зниження корисної продуктивності;
- переповнення адресних таблиць;
- широкомовні шторми;
- невикористання працездатних каналів за межами покривного дерева.

Дійсна робота присвячена розробці нових уніфікованих систем адресації мереж, які дозволяють вирішити зазначені проблеми і забезпечити гарантовану якість обслуговування, що зумовлює актуальність теми дослідження.

1 РОЗРОБКА СИСТЕМИ АДРЕСАЦІЇ E6

За теперішніх часів технологія Ethernet [33,37] стала домінуючою на ринку локальних мереж. Крім того, Ethernet витісняє інші технології в секторах корпоративних магістралей, мереж доступу і магістралей операторів зв'язку. Стандарти 1Гбіт/с и 10Гбіт/с Ethernet дозволяють використовувати її там, де традиційно застосовувалась технологія PDH/SDH (STM). Особливо швидкість передачі 10Гбіт/с зручна при реалізації DWDM магістралей [25]; вона відповідає максимальній швидкості передачі для окремої довжини хвилі та дозволяє забезпечити передачу потоку 400Гбіт/с по 40 доступних довжинах хвиль DWDM. Стандарти Ethernet широко застосовані в бездротових мережах [6,30]. Отже Ethernet стає універсальною технологією канального рівня, поступово витісняючи інші технології.

В більшості телекомунікаційних мереж поверх Ethernet працюють протоколи сім'ї TCP/IP [24,31]. Популярність цієї сім'ї завойовано придатністю саме протоколів мережного-сеансового рівнів IP [64], TCP [65], UDP [63] інтегрувати різні канальні технології і таким чином об'єднувати різні мережі канального рівня. Багаж застосувань які виросли в середовищі TCP/IP, таких як всесвітня павутина HTTP, електронна пошта SMTP, телефонія VoIP, являє собою основну цінність вказаної сім'ї протоколів [12,24,25,31].

За обставин коли кінцеві (абонентські) та локальні мережі, а також магістралі розбудовано на основі однієї та такої ж самої технології (Ethernet), протоколи мережного-сеансового рівнів, а саме TCP, IP, в сім'ї TCP/IP стають надлишковими. Практично всі задачі яки вирішують ці протоколи можуть бути вирішено за допомогою протоколів Ethernet. Адресацію хостів може бути успішно виконано за допомогою MAC-адрес Ethernet; фрагментація при тотальному застосуванні Ethernet не потребується; передачу даних з встановленням з'єднання алгоритмами ковзного вікна може бути реалізовано Ethernet LLC2 [51]. Необхідними доповненнями до стандартних можливостей Ethernet є лише: структурування MAC-адрес для маршрутизації, додання номерів портів в стандартні заголовки Ethernet. Отже від мережно-сеансовоих протоколів сім'ї TCP/IP залишається лише пара номерів портів для адресації мережних застосувань. Ліквідуються надлишкові заголовки TCP, IP, мінімальна довжина яких складає 40 байтів, а також допоміжні протоколи, такі як ARP/RARP [62].

Метою дійсного розділу є розробка систем адресації та схем реалізації глобальних мереж на основі тотального застосування Ethernet, анулювання протоколів TCP, IP з повним збереженням інтерфейсів прикладного рівня.

1.1 Попередній опис схеми адресації E6 та її варіантів

Розглянемо необхідність застосування подвійної адресації (MAC-адреса. IP-адреса) в мережі, яка повністю побудована на основі технології Ethernet. Спочатку вивчимо більш детально вказані адреси. MAC-адреса інтерфейсу Ethernet складається з 6 байтів: перші 3 байти зберігають код фірми-виробника, 3 наступних байти являють собою індивідуальний номер пристрою. Ethernet передбачає також широкомовну адресу, яка складається з двійкових одиниць та групові адреси. Апаратні засоби (мережні адаптери, комутатори) не інтерпретують індивідуальні адреси, а використовують їх лише як ключову інформацію для порівняння (повний збіг) в таблицях комутації. Отже застосування індивідуальних неструктурованих адрес приводить до необхідності використання таблиць, які містять перелік всіх індивідуальних адрес, що є практично неможливим для глобальних мереж. Ще одним негативним наслідком застосування неструктурованих адрес є широкомовні шторми, які суттєво знижують продуктивність глобальних мереж.

Слід зазначити, що в теперішній час практично всі інтерфейси Ethernet підтримують можливість програмного призначення MAC-адреси, що є однією з важливіших передумов для розбудови глобальних мереж. Крім того, практично всі дротові сучасні мережі Ethernet мікросегментовані: до кожного порту комутатора (маршрутизатора) підключений тільки один термінальний/мережний пристрій.

Перевагою IP-адресації є структурування адреси за допомогою розподілу на адресу IP-мережі та адресу хоста в мережі. IP-адреса складається з 4 байтів. В теперішній час в безкласовій системі адресації CIDR [44,66] найбільш простим способом опису структури є визначення кількості бітів мережної адреси:

194.46.88.237/12.
що відповідає класичному способу:
IP-адреса: 194.46.88.237, маска мережі: 255.240.0.0

Структурована адреса дозволяє використовувати в таблицях маршрутизації один запис для всієї IP-мережі і у такий спосіб скоротити розмір таблиць маршрутизації. Ще однією можливістю для скорочення розмірів таблиць маршрутизації є агрегація адрес декількох IP-мереж під спільною маскою, що є достатньо ефективним засобом при призначенні IP-адрес, що є близьким до оптимального [41,50]. Наприклад IP-мережі

194.224.0.0/12 та 194.208.0.0/12

можна агрегувати в IP-мережу

194.192.0.0/10

Пропонуємо ввести структурування MAC-адрес для повного відмовлення від використання IP-адрес. Дійсно, поле коду виробника не інтерпретується апаратно; MAC-адресу сучасних інтерфейсів Ethernet можна призначити програмно. Отже можна використати всі 6 байтів адреси (за виключенням двох перших службових бітів) для створення нової системи адресації Ethernet інтерфейсів. З іншого боку отриману адресу можна розглядати як CIDR IP адресу [44], яка розширена до 6 байтів.

В результаті попереднього аналізу обрано три наступних перспективних схеми адресації:

- **E6.** Використання всіх 6-ти байтів в якості мережної адреси з додатковим її розділенням на адресу мережі та адресу хоста мережі за допомогою маски (кількості бітів мережної адреси).
- **E6-4.** Збереження існуючої IP-адреси в останніх 4 байтах MAC-адреси; перші два байта заповнюються нулями та не використовуються.
- **E4P.** Збереження існуючої IP-адреси в останніх 4 байтах MAC-адреси і номера порту в перших 2-х байтах.

Вказані схемі адресації мають як свої переваги, так і недоліки:

Схема E6 дає можливість вирішити існуючу проблему дефіциту IP-адрес, збільшуючи кількість доступних адрес в $2^{14} \approx 16000$ разів. Вона не вимагає істотної зміни алгоритмів роботи інтерфейсів. Але виникає необхідність розміщення 4 байтів номерів портів відправника і одержувача або в існуючих заголовках кадрів Ethernet або в додатковому заголовку (формату скороченого UDP)**.** Крім того, цей спосіб вимагає перетворення системи DNS для використання 6-ти байтних адрес і модифікації прикладних інтерфейсів.

Схема E6-4 забезпечує переймання існуючої IP-адресації, вона не вимагає внесення змін в застосування і систему DNS та істотної зміни алгоритмів роботи інтерфейсів. Однак, як і в Схемі E6 виникає необхідність розміщення 4 байтів номерів портів відправника і одержувача або в існуючих заголовках кадрів Ethernet або в додатковому заголовку (формату скороченого UDP)**.**

Схема E4P не вимагає створення додаткових заголовків кадру Ethernet для номерів портів; вона дозволяє повністю зберегти існуючу систему IP-адресації. Однак вимагає істотної зміни алгоритмів роботи інтерфейсів і комутаторів (маршрутизаторів), які повинні ігнорувати 2 байта номера порту при порівнянні адрес.

1.2 Корисна модель Е6

Результати попереднього опису нової системи адресації Е6 в підрозділі 1.1 потребують подальшої деталізації та визначення у формі нової корисної моделі організації мережі, яка може бути трансформована у низку нових мережних стандартів.

Повну назву корисної моделі, на яку отримано відповідний патент [8], подано як: спосіб передачі даних в мережі із заміщенням мережного та транспортного рівнів універсальною технологією канального рівня.

1.2.1 Формула корисної моделі

1 Спосіб передачі даних в мережі із заміщенням мережного та транспортного рівнів універсальною технологією канального рівня, який включає використання стандартного формату заголовку кадру Ethernet з полями адрес довжиною в шість байтів, який **відрізняється** використанням на всіх рівнях еталонної моделі взаємодії відкритих систем єдиних мережних Е6-адрес довжиною в шість байтів, що надає можливість розміщення Е6-адрес замість MAC-адрес у заголовку кадру Ethernet, який **відрізняється** ієрархічною структурою Е6-адрес, яка складається з номеру мережі та номеру вузла мережі, при цьому запобігається необхідність передачі сигналів стосовно відображення адрес різних рівнів еталонної моделі, а також забезпечується можливість суттєвого скорочення адресних таблиць мережних пристроїв завдяки агрегації окремих адрес вузлів та адрес підмереж в адресу мережі наступного рівня ієрархії, що зумовлює можливість розбудови глобальних мереж з більшою кількістю підключених вузлів.

2 Спосіб за п.1, який **відрізняється** тим, що замість протоколів UDP та TCP використовуються апаратні можливості Ethernet LLC1 та LLC2 відповідно, замість протоколу IP використовується стандартний апаратний заголовок кадру Ethernet з Е6-адресами, який є незмінним у процесі доставки кадру до кінцевого вузла.

3 Спосіб за п.1, який **відрізняється** тим, що мережа розбудована із спеціальних комутуючих маршрутизаторів КМЕ6, які підключені один до одного та до кінцевих вузлів мережі, що дозволяє використовувати в КМЕ6 адресні таблиці з агрегацією адрес для вирішення задач маршрутизації та індивідуальні адреси для вирішення задач комутації, при цьому адреса інтерфейсу КМЕ6 задається тільки номером його порту, крім того КМЕ6 використовує тільки інформацію із стандартного заголовку кадру Ethernet, що дозволяє скоротити об’єм адресних таблиць та зменшити час ретрансляції кадру за рахунок зменшення розміру шини адреси та спрощення алгоритмів обробки кадру.

1.2.2 Опис корисної моделі

Запропонована корисна модель відноситься до техніки зв'язку, зокрема до процедур передачі даних в комп'ютерній мережі (КМ), цілком побудованій за однією технологією канального (та фізичного) рівня. Для фізичного передавання інформації використано технологію Ethernet [55] із спеціальною додатковою інтерпретацією фізичних адрес інтерфейсів пристроїв, яка визначає спосіб передачі даних у вигляді кадрів, метод доступу до середовища передавання, спосіб кодування інформації, модуляції електричного сигналу тощо.

Відомим способом передачі даних між двома вузлами КМ є використання мережних адрес (IP-адрес) [64] довжиною в чотири байта з ієрархічною структурою, які в процесі передачі пакетів інформації відображаються на фізичні адреси (MAC-адреси) [55] пристроїв довжиною в шість байтів. При цьому використання протоколів відображення адрес впливає як на послідовність передавання електричних сигналів, що переносять службову та корисну інформацію, так і на зміст цієї інформації.

Найближчим аналогом пропонованого способу є застосування стандартної інкапсуляції IP-Ethernet [49], яка використовує спеціальні протоколи ARP/RARP для прямого та зворотного відображення мережних IP-адрес на MAC-адреси фізичних пристроїв (їх мережних адаптерів). Вузол **A** з мережною адресою **IPA** та фізичною адресою **MACA** надсилає пакет даних вузлу **B** з мережною адресою **IPB** та фізичною адресою **MACB** через послідовність *k* проміжних вузлів (маршрутизаторів) **R1**, … , **Rk** з фізичними адресами **MAC1**, …, **MACk** у такий спосіб, що пакет інкапсулюється послідовно в Ethernet кадри між сусідніми маршрутизаторами із застосуванням пари адрес одержувача (**IPB**, **MACi**), де **MACi** – фізична адреса наступного маршрутизотора або кінцевого вузла **B**. Фізична адреса **MACi** поточного маршрутизотора або фізична адреса **MACB** кінцевого вузла **B** визначається початковим вузлом **A** або попереднім маршрутизатом **MACi-1** за допомогою спеціальних таблиць відображення протоколу ARP. Сім'я протоколів TCP/IP використовує додатково транспортні протоколи TCP [65] та UDP [63], які відповідають за адресацію застосовувань (клієнтів та серверів) на кінцевих вузлах **A** і **B**. Адресу застосування подано цілим числом, яке має назву номер порту. Протокол UDP забезпечує надсилання окремих пакетів без підтверджень. Протокол TCP виконую гарантовану доставку потоків (сегментів) даних через послідовність пакетів на основі процедури ковзного вікна [65].

Такий спосіб передачі даних є повністю виправданим для використання в телекомунікаційних TCP/IP-мережах, які застосовують різні технології канального та фізичного рівнів, однак є занадто надлишковим для використання у мережах цілком побудованих на

основі однієї універсальної технології канального (та фізичного) рівнів, наприклад Ethernet. Серед недоліків використання такого способу передачі даних у мережах, фізичні та канальні інтерфейси яких організовано тільки з використанням однієї технології (Ethernet), можна відзначити такі:

– необхідність подвійної адресації вузлів за допомогою мережних IP-адрес та фізичних MAC-адрес;

– необхідність використання додаткових протоколів відображення адрес, таких як ARP/RARP та відповідного додаткового програмного забезпечення або пристроїв;

– використання повільних процедур ковзного вікна протоколу TCP з великими таймаутами замість швидких процедур ковзного вікна технології канального рівня (LLC2 для Ethernet);

– надлишкові заголовки TCP та IP, які складають мінімально 40 байтів для кожного пакета даних, що особливо критично при передачі даних телефонії;

– використання складних пристроїв для маршрутизації пакетів у мережі, які забезпечують подвійну адресацію та відображення адрес, що знижує їх продуктивність;

– обмежена кількість мережних адрес (2^{32}), яких боргує у глобальних мережах, що вимагає застосування складних методів трансляції адрес.

Зазначені недоліки призводять до погіршення показників якості обслуговування (збільшення часу доставки пакету) та зниження продуктивності мереж, а іноді унеможливлює надання деяких послуг, наприклад послуг телефонії, які вимагають обмежений гарантований час доставки порівняльно невеликих пакетів (100 байтів), для яких додаткові заголовки TCP, IP є занадто надлишковими, крім того існуюче обмеження кількості мережних адрес перешкоджає розвитку глобальних мереж.

Поставлена задача полягає в розробці способу (моделі) передачі даних із використанням єдиної адресації мережних вузлів тільки за допомогою модифікованих MAC-адрес інтерфейсів технології Ethernet, що дозволяє позбавитися надлишкових протоколів TCP, UDP, IP і подвійної адресації вузлів мережі та забезпечує передачу даних за менший час та з використанням меншої кількості дій.

В запропонованому способі це досягається шляхом впровадження ієрархії MAC-адрес на основі їх розподілення на підмережі та програмного призначення адрес відповідним апаратним інтерфейсам.

Технічно задача вирішується шляхом впровадження нової єдиної адреси пристрою, яку названо E6-адресою, у форматі шості байтів існуючої MAC-адреси інтерфейсу Ethernet. Відповідно до існуючих стандартів [55] MAC-адреса Ethernet (Рис. 1.1, Табл. 1.1) не інтерпретується кінцевими та проміжними (комутатор, маршрутизатор) пристроями, а

використовується як натуральне число (до 2^{48}), що визначає унікальний номер пристрою (за винятком групових та широкомовних адрес зазначених двома першими бітами). Існуюче розподілення MAC-адрес на код виробника (перші 3 байта) та номер пристрою (останні 3 байта) не використовується при передачі даних. Крім того, сучасні адаптери Ethernet, комутатори і маршрутизатори дозволяють встановити програмним шляхом нову довільну MAC-адресу. Запропоновано встановити на всіх Ethernet інтерфейсах E6-адреси замість стандартних MAC-адрес. Адреса E6 (Рис. 1.2, Табл. 1.2) складається із адреси мережі та адреси вузла (хоста). Для додаткового застосування в комунікаційних пристроях використана кількість бітів адреси мережі, що традиційно назвемо маскою. Пропонується запис E6-адреси побайтно через крапку з поданням маски через риску, наприклад 42.53.64.25.172.48/44. Можливе застосування спеціальних адрес: E6-адреса з усіма бітами адреси вузла рівними нулю є спеціальною адресою усієї підмережі; E6-адреса з усіма бітами адреси вузла рівними одиниці є спеціальною широкомовною адресою, що адресує усі вузли підмережі. Визначена структура E6-адреси дозволяє агрегатувати окремі адреси або адреси підмереж під спільною маскою мережі, що надає можливість уникнути розростання кількості індивідуальних адрес пристроїв у адресних таблицях та організувати глобальну мережу з простором адрес, який перевищує простір IP-адрес в 2^{14} разів.

Для реалізації запропонованого способу передачі даних необхідне використання спеціальних комутуючих маршрутизаторів (КМЕ6) з інтерфейсами всіх портів за стандартам Ethernet, робоча процедура (**ПроцКМЕ6**) яких може бути подана у такий спосіб:

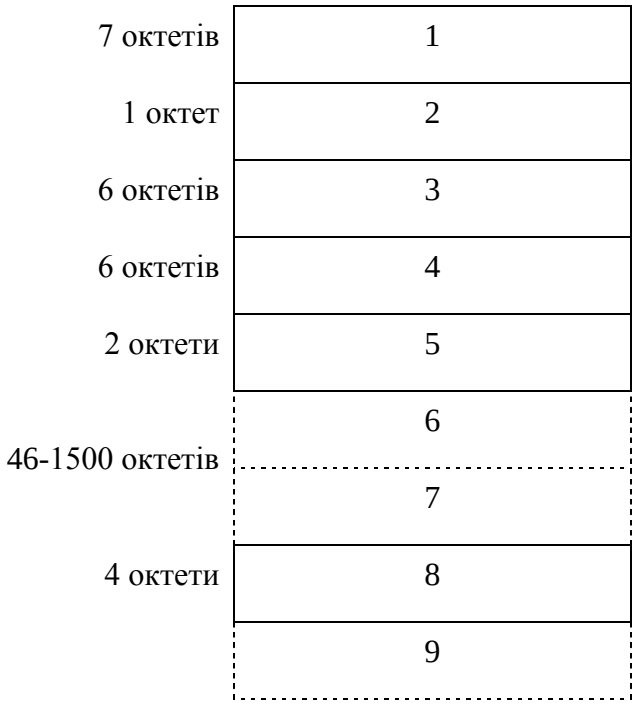


Рисунок 1.1 – Формат стандартного кадру Ethernet

Таблиця 1.1 – Умовні позначення до Рис. 1.1

1	преамбула
2	обмежник початку кадру
3	адреса одержувача кадру
4	адреса відправника кадру
5	довжина або тип
6	додаткові заголовки та дані кадру
7	заповнювач
8	контрольна послідовність кадру
9	розширення кадру

– прийняти кадр на певний порт КМЕ6 із застосуванням стандартних способів кодування сигналів Ethernet відповідними інтерфейсами;

– вилучити Е6-адресу одержувача з поля першої МАС-адреси кадру;

– знайти запис в адресній таблиці з маскою найбільшої довжини, що задовольняє Е6-адресі одержувача кадру, та найменшою метрикою;

– визначити номер порту призначення КМЕ6 із запису адресної таблиці;

– передати кадр з визначеного порту призначення КМЕ6 із застосуванням стандартних способів кодування сигналів Ethernet відповідними інтерфейсами.

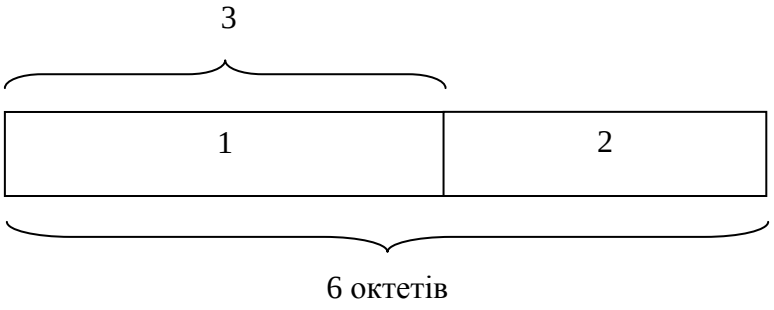


Рисунок 1.2 – Формат єдиної мережної адреси Е6

Таблиця 1.2 – Умовні позначення до Рис. 1.2

1	номер (адреса) мережі (підмережі)
2	номер (адреса) вузла в мережі
3	маска – кількість бітів адреси мережі

Запис адресною таблиці складається з E6-адреси, маски, номеру порту, метрики та додаткової інформації. Для ефективності роботи всі термінальні пристрої підключені до КМЕ6 мають E6-адреси під однією спільною маскою E6 мережі. Для визначення цієї мережі можливе застосування першого запису адресної таблиці з недійсним номером порту, наприклад 0 та метрикою 0. Можливе застосування запису 0.0.0.0.0.0/0 для адресації всіх невідомих E6-адрес.

Мережа передачі даних складається з підключених один до одного КМЕ6, та термінальних пристроїв, підключених до вільних портів КМЕ6. Процедура передачі даних від певного застосування **ТА** вузла **А** з адресою **Е6А** та номером порту **РА** певному застосуванню **РВ** вузла **В** з адресою **Е6В** та номером порту **РВ** подається у такий спосіб:

– використати E6-адреси замість IP- та MAC-адрес;

– на вузлі **А** обрати LLC1 Ethernet при визові UDP, обрати LLC2 Ethernet [51] при визові TCP;

– сформувати кадр (кадри) Ethernet з **Е6В** замість MAC-адреси одержувача та **Е6А** замість MAC-адреси відправника та додатковим заголовком HP2 (Рис. 1.3, Табл. 1.3), який містить пару номерів **РВ**, **РА** портів застосувань;

– передати кадр до підключеного КМЕ6 із застосуванням стандартних способів кодування сигналів Ethernet відповідними інтерфейсами;

– виконати доставку кадру через послідовність проміжних КМЕ6 згідно до **ПроцКМЕ6** поданою раніше;

– прийняти кадр в кінцевому вузлі **В** із застосуванням стандартних способів кодування сигналів Ethernet відповідними інтерфейсами;

– вилучити номери портів **РВ**, **РА** із додаткового заголовка HP2;

– при використанні LLC1 Ethernet передати дані кадру програмам застосування **ТВ**, які чекають на порту **РВ** пакет UDP, при використанні LLC2 Ethernet передати дані кадру програмам застосування **ТВ**, які чекають на порту **РВ** сегмент TCP.

Одним з можливих варіантів застосування запропонованого способу передачі даних у глобальних мережах є розробка додаткових протоколів доменних імен DNS-E6, динамічної маршрутизації RIP-E6, OSPF-E6, BGP-E6, автоматичного призначення E6-адрес вузлів DHCP-E6 аналогічних до відповідних стандартних протоколів сім'ї TCP/IP для забезпечення використання імен замість E6-адрес, динамічної розбудови адресних таблиць КМЕ6, автоматичного призначення E6-адрес підмережам та вузлам відповідно. При використанні застосуваннями прикладного рівня стандартних доменних імен запропонований спосіб передачі даних на основі E6-адрес буде прозорим для кінцевого користувача мереж. Крім того, запропонований спосіб вимагає лише незначних змін у програмах прикладного рівня пов'язаних з розширенням адрес з 4 до 6 байтів.

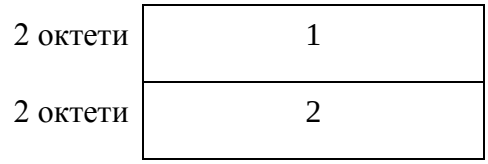


Рисунок 1.3 – Формат додаткового заголовку HP2

Таблиця 1.3 – Умовні позначення до Рис. 1.3

1	номер порту застосування одержувача
2	номер порту застосування відправника

Переваги запропонованого способу передачі даних полягають в:

– застосуванні єдиної системи Е6-адрес у всій мережі, що дозволяє уникнути подвійної адресації та процедур відображення адрес при інкапсуляції пакетів в кадри Ethernet на телекомунікаційних пристроях;

– можливості уникнути застосування іншої інформації окрім заголовку кадрів Ethernet (який залишається незмінним) при маршрутизації пакетів на телекомунікаційних пристроях (КМЕ6);

– спрощенні алгоритмів роботи телекомунікаційних пристроїв (КМЕ6) завдяки використанню лише номера фізичного порту як єдиного ідентифікатора інтерфейсу;

– скороченні об’єму службової інформації заголовків пакетів мінімально на 40 байтів завдяки фактичному анулюванню протоколів TCP, UDP, IP;

– зменшенні кількості операцій обробки пакета в процесі його доставки та відповідно зменшенні часу передачі пакету через мережу між кінцевими вузлами, що зумовлює підвищення ефективності обміну інформацією в мережі;

– уникнути використання повільних процедур ковзного вікна протоколу TCP завдяки безпосередньому використанню аналогічних швидких процедур канального рівня LLC2 Ethernet і у такий спосіб прискорити передачу потоків даних;

– розширити простір адрес мережі в 2^{14} рази порівняльне з IP-адресами.

1.3 Стек протоколів E6

1.3.1 Інтерфейси прикладного рівня

Запропоновано звести до мінімуму модифікації інтерфейсів прикладного рівня для забезпечення роботи всіх існуючих мережних застосувань у такий спосіб що зміни стеку протоколів будуть непомітними для кінцевого користувача.

Без використання протоколів рівню подання інформації таких як SSL [24,25,31], інтерфейси прикладного рівня достатньо прості. Вони складаються в визначенні сокетів відправника та одержувача і номеру протоколу (UDP**,** TCP) в командах читання/запису для клієнтів/серверів і командах прослуховування сокету для серверів. Сокет складається із пари: IP-адреса, номер порту.

Номер протоколу достатньо просто моделювати використанням Ethernet LLC1 для протоколу UDP і Ethernet LLC2 для протоколу TCP, що забезпечує просту передачу кадрів без встановлення з'єднання в першому випадку і передачу кадрів із встановленням з'єднання та алгоритмом ковзного вікна у другому випадку.

Номер порту пропонуємо залишити незмінним для адресації мережних застосувань всередині хоста із збереженням всіх стандартних значень добре-відомих видів сервісу [67] і розподіленням на порти серверів та клієнтів.

IP-адреса також залишається незмінною при застосуванні Схем адресації E6-4, E4P з повним збереженням існуючої системи імен DNS.

Застосування 6-ти байтних адрес при використанні Схеми адресації E6 вимагає повної перекомпіляції всіх застосувань з однією модифікацією коду, яка складається в збільшенні кількості байтів адреси. Крім того потрібна відповідна модифікація існуючих таблиць системи DNS [61].

В простішому випадку можна розмістити існуючі IP-адреси в останні 4 байти E6-адреси, доповнивши його певним номером мережі, наприклад 1.0 зліва до 6 байтів, що дозволяє сприйняти повністю існуючу де-факто систему призначення IP-адрес. Надалі можливе використання повного адресного простору, що надає можливість розширити мережу до розміру порівняного з IPv6 [39]. Модифікація таблиць системи DNS [61] може бути виконано нескладними конверторами [18], що доповнюють адресу початковим номером мережі зліва: 1.0.194.46.88.237.

Однак уявляється доцільним виконати оптимізацію [41] розподілення адресного простору при переході на нову систему адресації E6 з урахуванням топології маршрутизації [50] і перспектив подальшого розширення.

1.3.2 Опис стеку E6

Стандартний стек протоколів сім'ї TCP/IP [70] модифіковано у такий спосіб, що множина протоколів рівнів мережний-сеансовий замінено єдиним рівнем Узгодження E6/E6-4/E4P (Рис. 1.4). Для простоти рівень подання інформації (SSL та інші протоколи) вилучені із розгляду, але вони також можуть бути інтегровані в Схеми E6/E6-4/E4P.

Функції рівня Узгодження E6:

- вибір LLC1 замість UDP, LLC2 замість TCP;
- розміщення E6-адреси відправника в MAC-адресі відправника;
- розміщення номеру порту відправника в додатковому заголовку HP2 (Рис. 1.3);
- розміщення E6-адреси одержувача в MAC-адресі одержувача;
- розміщення номеру порту одержувача в в додатковому заголовку HP2;
- розміщення інформації QoS в полі пріоритету заголовка 802.1p/Q.

Функції рівня Узгодження E6-4:

- вибір LLC0 замість UDP, LLC2 замість TCP;
- розміщення IP-адреси відправника в останніх 4 байтах MAC-адреси відправника, заповнення перших 2 байтів нулями;
- розміщення номеру порту відправника в додатковому заголовку HP2;
- розміщення IP-адреси одержувача в останніх 4 байтах MAC-адреси одержувача, заповнення перших 2 байтів нулями;
- розміщення номеру порту одержувача в в додатковому заголовку HP2;
- розміщення інформації QoS в полі пріоритету заголовка 802.1p/Q.

<table>
<tr><td>OSI-ISO</td><td colspan="2">TCP/IP</td><td rowspan="6">IP
|
|
|
|
|
|
|
|
|
V
MAC,IP</td><td colspan="2">E6</td><td rowspan="6">E6
|
|
|
|
|
|
|
|
|
V
E6</td></tr>
<tr><td>Прикладний</td><td colspan="2">HTTP, SMTP, VoIP ...</td><td colspan="2">HTTP, SMTP, VoIP ...</td></tr>
<tr><td>Сеансовий</td><td rowspan="2">TCP</td><td></td><td colspan="2" rowspan="3">E6 Узгодження</td></tr>
<tr><td>Транспортний</td><td>UDP</td></tr>
<tr><td>Мережний</td><td colspan="2">IP</td></tr>
<tr><td>Канальний</td><td colspan="2">Ethernet</td><td colspan="2">E6 Ethernet</td></tr>
</table>

Рисунок 1.4 – Стеки протоколів: OSI-ISO, TCP/IP, E6/E6-4/E4P

Функції рівня Узгодження E4P:

- вибір LLC0 замість UDP, LLC2 замість TCP;
- розміщення IP-адреси відправника в перших 4 байтах MAC-адреси відправника;
- розміщення номеру порту відправника в останніх 2 байтах MAC-адрес відправника;
- розміщення IP-адреси одержувача в перших 4 байтах MAC-адреси одержувача;
- розміщення номеру порту одержувача в останніх 2 байтах MAC-адрес одержувача;
- розміщення інформації QoS в полі пріоритету заголовка 802.1p/Q.

В простішому випадку заголовок HP2 в Схемах E6, E6-4 може бути добавлений за допомогою звичайної інкапсуляції як і при реалізації стеку TCP/IP. Можливе також його розміщення в додаткових заголовках Ethernet, наприклад SNAP, TCI.

Для реалізації вказаних схем обрано формати кадрів IEEE 802.3/LLC [51] та 802.lp/Q [53,54], які дозволяють реалізувати 3 рівня LLC, передбачених стандартами, а також вказати параметри QoS у заголовку TCI віртуальних мереж. Зазначимо, що при цьому максимальна довжина поля даних кадру (1500) скорочується на 3-7 байтів.

Слід відзначити, що класифікація стеків Схем E6/E6-4/E4P виконано с точки зору заміщення відповідних рівнів стеку TCP/IP. Однак множина існуючих підрівнів канального рівня Ethernet створила класифікаційні складності де-факто. При більш точній класифікації рівень Узгодження E6/E6-4/E4P може бути схарактеризований як транспортний; підрівень Ethernet MAC – як мережний рівень, підрівні Ethernet LLC1,2 – як транспортні та сеансові.

1.4 Побудова мережі на основі схеми адресації E6

Схема E6 не вимагає зміни алгоритмів роботи Ethernet інтерфейсів. Однак, потрібно інкапсуляція заголовка HP2, який складається з 4 байтів. Крім того, у загальному випадку потрібно застосування спеціальних комутуючих маршрутизаторів E6.

1.4.1 Мережні інтерфейси

Логічний інтерфейс представлений вказівкою E6-адреси і маски E6-мережі. Замість комутаторів і маршрутизаторів використовується комутуючий маршрутизатор E6; можливо також застосування звичайних комутаторів на периферії мережі, що буде розглянуто нижче. У випадку тотального застосування комутуючих маршрутизаторів E6 вказівка E6-адреси маршрутизатора по-умовчанню є зайвою.

Фізичний інтерфейс конфігурується як MAC-адреса, що являє собою копію адреси E6. Модифікація роботи інтерфейсів не потрібно. Програмне забезпечення Узгодження E6 копіює MAC-адресу в E6 адресу, витягає номера портів з додаткового заголовка HP2 і передає інформацію відповідному застосуванню.

1.4.2 Алгоритм роботи комутуючого маршрутизатора E6

Сполучений комутуючий маршрутизатор E6 (КМЕ6) виконує безпосередню комутацію кадру в тому випадку, якщо пристрій призначення підключений до його власного інтерфейсу, і визначення номера порту наступного хопу (шлюзу) у противному випадку. Різницею формату таблиць у порівнянні з класичними таблицями комутації є наявність як записів про індивідуальний хост так і записів про мережі:

Адреса пристрою/мережі		Номер порту
E6-адреса	Число бітів адреси мережі (маска)	

Єдиний формат забезпечується вказівкою кількості бітів мережі в адресі. Якщо зазначене число бітів мережі дорівнює 48, то підключений термінальний пристрій, у противному випадку підключена певна мережа, у яку можна потрапити через зазначений порт (через певну послідовність комутуючих маршрутизаторів). Як і в стандартних алгоритмах маршрутизації, перевага віддається запису з найбільш специфічною маскою, таким чином, гарантується доставка безпосередньо підключеному пристрою. При використанні мікросегментованої Ethernet зазначений номер порту однозначно задає наступний хоп і не вимагає його специфікації парою: адреса інтерфейсу, адреса хопу (шлюзу).

У таблиці можуть бути включені додаткові поля метрик для вибору наступного хопу при наявності декількох альтернативних маршрутів. Таблиці можуть задаватися статично. Крім того, можлива адаптація стандартних алгоритмів динамічної маршрутизації RIP [47,59], OSPF [25,31] та інших для побудови таблиць. Адаптація алгоритмів не є складною. Потрібно, лише спрощення специфікацій адреси шлюзу при заміні його номером відповідного порту.

Для забезпечення цілісності системи маршрутизації доцільне призначення підмережі на КМЕ6. Власна підмережа може бути представлена записом таблиці з указівкою неіснуючого номера порту, рівного 0. Крім того, для віддаленого конфігурування КМЕ6 потрібно призначення йому певної адреси. Для простоти можна вважати, що адреса 1 власної мережі призначена комутуючому маршрутизатору.

1.4.3 Схема доставки пакету в Е6 мережі

Схема доставки пакету (кадру) має вигляд зображений на Рис. 1,5. На відміну від схеми доставки IP пакету (Рис. 1.6.) пара Е6 адрес одержувача та відправника (E6Y,E6X) є незмінною у процесі доставки в той час, як використання IPoverEthernet [25,49] вимагає знаходження нових MAC адрес шлюзу на кожному хопі, а також деінкапсуляції IP заголовку кадру для прийняття маршрутного рішення.

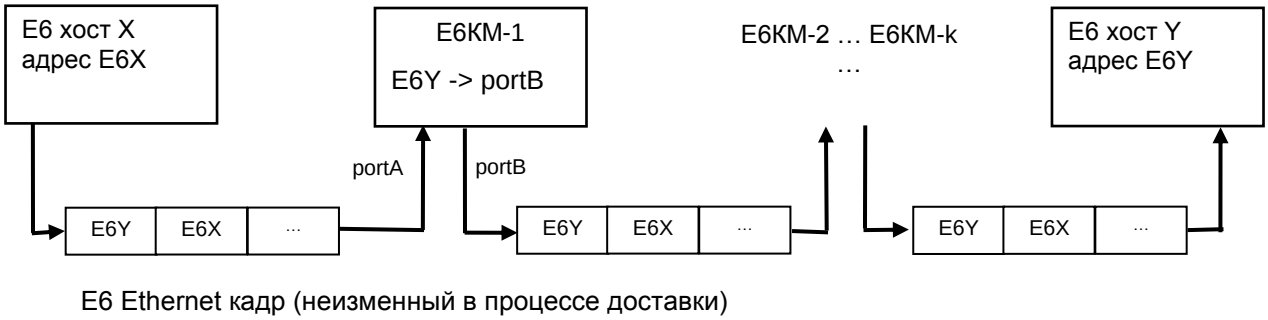


Рисунок 1.5 – Схема доставки Е6 пакета

Зазначені процеси знаходження MAC адреси шлюзу вимагають не тільки застосування внутрішніх таблиці відповідного пристрою, але також виконання ARP-запитів у разі відсутності запису у таблиці. Вказаний процес виконується не тільки одноразово для кожної нової IP адреси шлюзу, а також періодично у разі анулювання запису за таймером старіння, що вносить важко передбачені затримки до часу доставки IP пакету та може суттєво знижувати якість обслуговування. Затримка може бути достатньо великою порівняльне до продуктивності пристрою через те що необхідне дворазове передавання службового кадру ARP (запит та відповідь) через відповідний сегмент мережі.

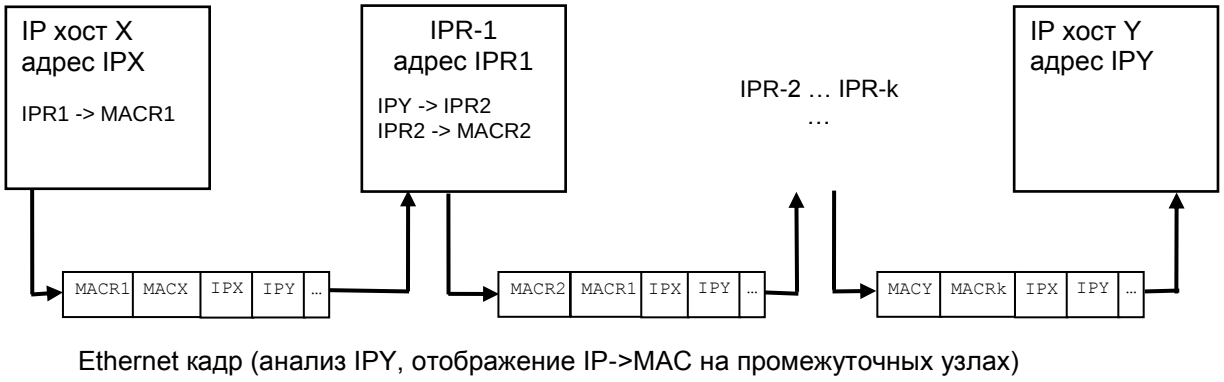


Рисунок 1.6 – Схема доставки IP пакета

Отже, схема доставки кадру у E6 мережі є більш досконалою порівняльне до стандартної технології IPoverEthernet. Відповідні оцінки часу доставки пакету будуть отримані у наступному розділі.

1.4.4 Приклад реалізації мережі на основі Схеми адресації E6

Структурна схема мережі, що складається з 5 КМЕ6, 64 термінальних пристроїв з підключенням до глобальної мережі представлена на Рис. 1.7.

Зауважимо, що через те що E6 адреса інтерпретується також як розширена до 6 байтів CIDR IP-адреса [44,66], адресу з усіма нульовими бітами поля номеру хоста зарезервовано як мережну адресу, а адресу з усіма одиницями – як широкомовну, кількість доступних хостів для мереж з різною довжиною поля номеру хоста вказано в Табл. 2.4.

Таблиця 1.4 – Кількість термінальних пристроїв E6 підмереж

Маска	Бітів адреси хоста	Кількість пристроїв	Кількість термінальних пристроїв
46	2	2	1
45	3	6	5
44	4	14	13
43	5	30	29
42	6	62	61
…	…	…	…
4	44	17592186044414	17592186044413
3	45	35184372088830	35184372088829
2	46	70368744177662	70368744177661

Для кожного КМЕ6 на схемі зазначений його номер, E6 адреса власної мережі з маскою, а також номера портів, підписані на лініях зв'язку. Крім того, зазначені E6 адреси термінальних пристроїв. Зауважимо, що використано КМЕ6 з 13 портами для E6 мереж під маскою 44; 4 біти адреси хоста дають 16 значень, з них 0000 – для адресації підмережі, 1111 – широкомовна, 0001 – власна адреса КМЕ6.

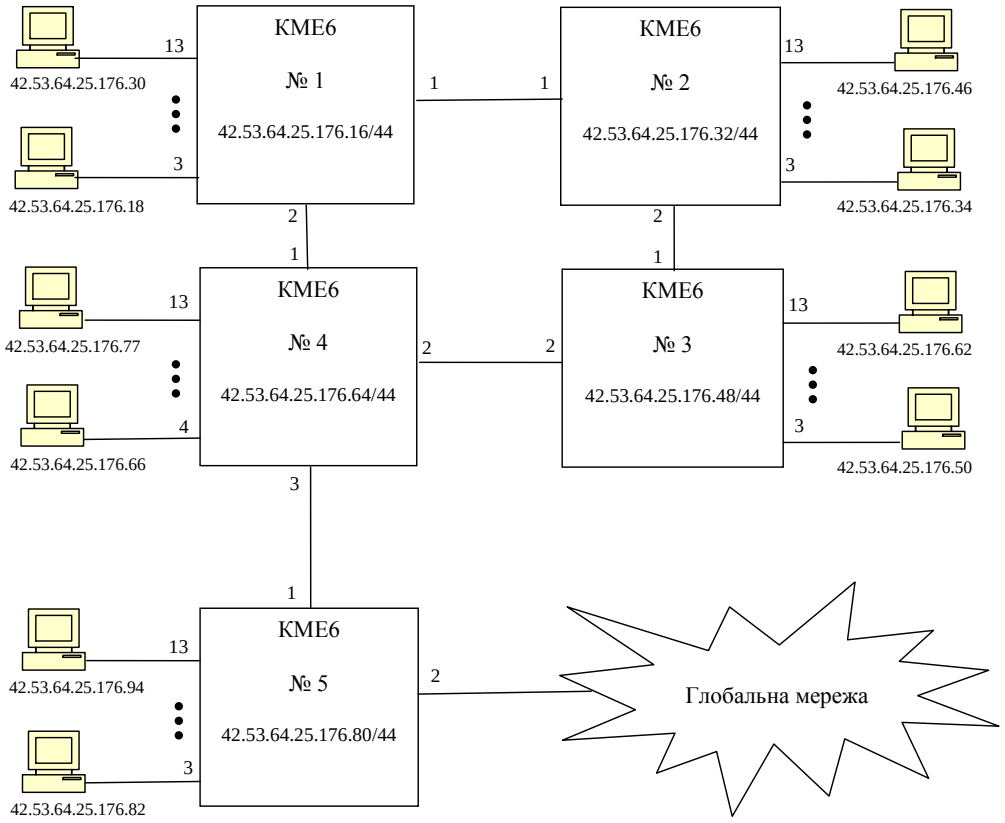


Рисунок 1.7 – Структурна схема Е6 мережі

Побудуємо таблиці комутації/маршрутизації (Табл. 2.5) для КМЕ6 мережі, поданої на Рис. 1.7. До таблиці долучене значення метрики, що у припущенні рівної швидкості портів може бути в найпростішому випадку оцінено як кількість проміжних ліній зв'язку (число хопів).

Перший рядок кожної таблиці задає власну мережу КМЕ6 за допомогою виділеного для цих цілей номера псевдопорту 0; потім йде таблиця комутації для безпосередньо підключених термінальних пристроїв з Е6 адресами під маскою 48; потім подано таблицю маршрутизації. Альтернативні маршрути рівної метрики не включені в таблиці. Зауважимо, що для маршруту по-умовчанню 0.0.0.0.0.0 у всіх таблицях зазначена однакова метрика, що перевищує метрику для усіх власних Е6-мереж.

За допомогою ручного трасування нескладно перевірити доставку кожного кадру по призначенню для всіх пар Е6 адрес відправника й одержувача для термінальних пристроїв. Таблиці задані статично, але їхня побудова може бути також виконано модифікованими протоколами RIP та OSPF.

Таблиця 1.5 – Таблиці комутації/маршрутизації КМЕ6

КМЕ № 1

Адреса	Маска	Порт	Метрика
42.53.64.25.172.16	44	0	0
42.53.64.25.172.18	48	3	0
…	…	…	…
42.53.64.25.172.30	48	13	0
42.53.64.25.172.32	44	1	1
42.53.64.25.172.64	44	2	1
42.53.64.25.172.48	44	2	2
42.53.64.25.172.80	44	2	2
0.0.0.0.0.0	0	2	5

КМЕ № 2

Адреса	Маска	Порт	Метрика
42.53.64.25.172.32	44	0	0
42.53.64.25.172.34	48	3	0
…	…	…	…
42.53.64.25.172.46	48	13	0
42.53.64.25.172.16	44	1	1
42.53.64.25.172.48	44	2	1
42.53.64.25.172.64	44	2	2
42.53.64.25.172.80	44	2	2
0.0.0.0.0.0	0	2	5

КМЕ № 3

Адреса	Маска	Порт	Метрика
42.53.64.25.172.48	44	0	0
42.53.64.25.172.50	48	3	0
…	…	…	…
42.53.64.25.172.62	48	13	0
42.53.64.25.172.32	44	1	1
42.53.64.25.172.64	44	2	1
42.53.64.25.172.16	44	1	2
42.53.64.25.172.80	44	2	2
0.0.0.0.0.0	0	2	5

КМЕ № 4

Адреса	Маска	Порт	Метрика
42.53.64.25.172.64	44	0	0
42.53.64.25.172.66	48	4	0
…	…	…	…
42.53.64.25.172.77	48	13	0
42.53.64.25.172.16	44	1	1
42.53.64.25.172.48	44	2	1
42.53.64.25.172.80	44	3	1
42.53.64.25.172.32	44	2	2
0.0.0.0.0.0	0	2	5

КМЕ № 5

Адреса	Маска	Порт	Метрика
42.53.64.25.172.80	44	0	0
42.53.64.25.172.82	48	3	0
…	…	…	…
42.53.64.25.172.94	48	13	0
42.53.64.25.172.64	44	1	1
42.53.64.25.172.16	44	1	2
42.53.64.25.172.48	44	1	2
42.53.64.25.172.32	44	1	3
0.0.0.0.0.0	0	2	5

1.4.5 Напрямки практичної реалізація E6 мережі

1 Призначення адрес

У найпростішому випадку призначення E6 адрес інтерфейсів та підмереж може бути виконане вручну. Однак для великих мереж такий спосіб може виявитися трудомістким. Уявляється доцільним розробка спеціального протоколу E6DCP, аналогічного по призначенню до DHCP [40], але який істотно відрізняється від нього. Найбільш ефективним уявляється автоматичне призначення E6 підмереж кожному КМЕ6, а потім КМЕ6 виконує призначення E6 адрес безпосередньо підключеним термінальним пристроям. Доповнення існуючих стандартів Ethernet протоколом HCP, що призначає адреси хостам дозволить анулювати недоліки Ethernet, пов'язані із широкомовними штормами. При підключенні хост передає спеціальний кадр запиту адреси, КМЕ6 задовольняє його запит з адресного простору власної E6 підмережі за допомогою спеціального кадру призначення адреси.

2 Стек протоколів

Стек протоколів традиційно має певну програмну реалізацію у ядрі відповідної операційної системи. Стек E6 може бути реалізований як додатковий модуль певної операційної системи. Завдяки гнучкості програмного підходу можлива інтеграція стеку E6 практично до всіх існуючих операційних систем. Найбільш простою є реалізація стеку E6 в операційній системі з відкритим вихідним кодом, такій як ОС UNIX [2,18,28]. Мінімальна модифікація прикладних інтерфейсів яка полягає у розширенні поля адреси з 4 до 6 байтів,

дозволяє інтегрувати до стеку E6 практично все існуюче прикладне програмне забезпечення стеку протоколів TCP/IP [48].

3 Мережні пристрої

Зауважимо, що на периферії мережі можливе застосування звичайних комутаторів Ethernet однак робота мережі може бути неефективної через неминуче широкомовлення з кожною появою адреси, що не міститься в комутованій підмережі. Широкомовлення забезпечить доставку кадру з невідомою E6 адресою призначення першому КМЕ6, до якого підключена комутована мережа, але розміри таблиць комутації будуть при цьому зростати.

Аналогічна проблема в мережах TCP/IP-Ethernet вирішена за допомогою вказівки адреси шлюзу по-умовчанню на кожнім хості [25]. Пропонується незначно модифікувати периферійні комутатори Ethernet для E6 мереж. Комутатор повинний знати адресу комутованої E6 підмережі і виконувати широкомовлення тільки для E6 адрес цієї підмережі. Для невідомих адрес, що не належать комутованій E6 підмережі, повинна бути забезпечена доставка кадрів до КМЕ6. Для цих цілей потрібно вказати лише виділений порт, що знаходиться на найкоротшому маршруті до КМЕ6. У багатьох випадках такі модифікації можуть бути виконані за допомогою спеціальних патчів мікропрограм комутатора.

КМЕ6 можуть бути реалізовані на основі існуючих маршрутизаторів. Потрібно модифікація використовуваних таблиць, що складається в розширенні адрес до 6 байтів і специфікації шлюзу номером порту. Однак найбільша ефективність побудови мережі може бути забезпечена спеціально спроектованими комутуючими маршрутизаторами E6.

4 Протоколи маршрутизації

Схема адресації E6 передбачає відмовлення від широкомовлення характерного для Ethernet у процесах доставки кадру. Адресні таблиці E6 повинні формуватися або вручну або за допомогою протоколів динамічної маршрутизації. Вважаючи схожість IP та E6 адрес (IP CIDR розширене з 4 до 6 байтів) пропонується адаптація відповідних протоколів IP маршрутизації [24,31] із істотним спрощенням завдяки використанню тільки двохточечних ліній зв’язку.

5 Шлюзування з мережами інших технологій

Існування E6 мережі незалежної від інших існуючих мереж не є реалістичним підходом завдяки великому об’єму інформаційних ресурсів нагромадженому особливо в мережі Інтернет та корпоративних мережах технології TCP/IP. Саме тому створення шлюзів, які забезпечать інформаційний обмін з мережами інших технологій, є однією з найважливіших задач практичного впровадження E6 мереж.

1.5 Особливості побудови мереж на основі Схем адресації E6-4/E4P

Схема E6-4 являє собою певний компроміс між перевагами використання єдиної мережної адреси и необхідними змінами в роботі програмних та апаратних засобів. Вона не дає можливості розширення адресного простору у порівнянні з IPv4 адресацію, але зберігає повне переймання роботи прикладного рівня і вимагає мінімальних змін протоколів динамічної маршрутизації.

Принципи побудови мереж залишаються тими же, що і для схеми E6, але E6-4 адреса інтерпретується як IP-адреса (32 біта) в роботі застосувань, програмного забезпечення DNS, протоколів динамічної маршрутизації, та як Ethernet MAC-адреса в роботі інтерфейсів.

Схема E4P відрізняється від схеми E6-4 тільки упаковкою номера порту в MAC-адресу і ліквідацією додаткового заголовка HP2, що економить 4 байта в довжині кожного кадру. Однак потрібна зміна алгоритмів роботи Ethernet інтерфейсів, що може бути виконано в багатьох випадках за допомогою спеціальних програмних патчів, які модифікують мікропрограми пристроїв. Однак привабливістю схеми E4P, як було зазначено раніше, залишається повне збереження інтерфейсів і програмного забезпечення прикладного рівня, існуючої IP-адресації в середовищі E4P.

Фізичний інтерфейс конфігурується як довільна MAC-адреса формату E4P, що містить в старших 4 байтах IP-адресу хоста. Для визначеності можна заповнити 2 байта номера порту нулями. Наприклад, для IP адреси 194.46.88.237 отримуємо 194.46.88.237.0.0.

Алгоритм роботи інтерфейсу модифіковано у такий спосіб, що довільна MAC-адреса одержувача, яка задовольняє мережі x1.x2.x3.x4.x5.x6/32 інтерпретується як своя власна, відповідний кадр приймається і передається в неінтерпретованому вигляді програмному забезпеченню Узгодження E4P. Програмне забезпечення Узгодження E4P вилучає номер порту та IP-адресу і передає інформацію відповідному застосуванню.

1.6 Дистанційно-векторний протокол динамічної маршрутизації E6

Запропоновано виконати адаптацію відомих алгоритмів (протоколів) динамічної маршрутизації, в основному стеку протоколів TCP/IP, в E6 мережі. Вибір обумовлений ієрархічною структурою E6 адреси, аналогічної до IP адреси. Альтернатива пасивного прослуховування і широкомовлення, прийнята в Ethernet (і PBB), відкинута, тому що вона стрибкоподібно знижує корисну продуктивність мережі і приводить до тимчасових перевантажень (розділ 4).

Принципи адаптації алгоритмів динамічної маршрутизації TCP/IP в E6 мережі можна сформулювати в такий спосіб:

– розширити поле адреси з 4 до 6 байтів і збільшити припустимий розмір маски з 32 до 48 бітів;

– замінити IP-адресу інтерфейсу й адресу шлюзу номером порту КМЕ6;

– модифікувати формати повідомлень і алгоритми їхнього формування, передачі, обробки.

Для початкового розвитку E6 мереж найбільш необхідною є адаптація дистанційно-векторного протоколу динамічної маршрутизації RIP [47,59], який використовується на периферії мережі. Робота протоколу заснована на періодичному розсиланні повної таблиці маршрутизації сусідам (з періодом uta) і негайним (з періодом tuta) розсиланням тригерних змін при відновленні маршрутів. Для ліквідації циклічних маршрутів використаний метод розщеплення горизонту, при якому відновлення не відправляється сусіду, від якого воно отримано.

Можливе використання складних метрик, але в більшості випадків як метрику розглядають кількість проміжних вузлів (хопів). Вибирають максимальне значення метрики INFINITY, що обмежує довжину припустимого маршруту. Мережі з метрикою INFINITY (і більше) вважають недосяжними.

Повідомлення E6-RIP має наступний формат:

(operation, e6nw, metric),

де operation представляє одну з операцій: request – запит, response – відповідь. Регулярні відновлення відправляються за таймером відновлень uta у форматі response навіть якщо не було відповідного запиту.

У таблиці маршрутизації виділені необхідні для роботи протоколу опції:

(e6nw, metric, port, chg, ta, gcta),

де chg – ознака змін, ta – таймаут старіння запису, gcta – таймаут збору сміття.

Таймаут ta переустановлюється при одержанні маршруту від сусідів; після закінчення ta запис відмічається як недосяжна мережа метрикою INFINITY і встановлюється gcta. По витіканню gcta запис вилучається з таблиці.

При одержанні відновлення метрика збільшується на вартість відповідного каналу (на одиницю). Новий маршрут до досяжної мережі додається в таблицю; новий маршрут до

недосяжної мережі (metric=INFINITY) ігнорується. Обробка відновлень для відомої мережі виконується в такий спосіб:

– замінити запис при меншій метриці;

– при одержанні INFINITY для запису з метрикою менше INFINITY, встановити метрику INFINITY і запустити gcta;

– ігнорувати одержання INFINITY для запису з метрикою INFINITY;

– ігнорувати відновлення з більшою метрикою (чи рівною метрикою й іншим номером порту);

– при одержанні відновлення з рівною метрикою і тим же номером порту переустановити ta.

Ознака змін chg використовується для наступного розсилання тригерних змін; вона установлюється при додаванні запису або зміні метрики. Опцією є випереджальне збереження альтернативного маршруту при рівній метриці і таймауті ta що минає.

Стандарт RIP [59] рекомендує наступні величини таймерів: uta=30 с., ta=180 с., gcta=120 с., tuta=1-5 с., які є предметом досліджень для E6-RIP. В Розділі 3 досліджувалася опція розщеплення горизонту з отруєним реверсом, коли відновлення перенаправляється відправнику з метрикою INFINITY. Крім того, досліджувалося саме значення INFINITY (стандартна величина дорівнює 16).

У розділі 3 описи протоколу представлені мовою розфарбованих сітей Петрі [58], що дозволяє безпосередньо використовувати їх у моделях мереж в середовищі CPN Tools [16,35], а також у подальшій програмно-апаратній реалізації КМЕ6.

Висновки до 1 розділу

1. Запропоновано нову систему E6 адресації мереж, та її модифікації E6-4, E4P, які дозволяють уникнути відображення адрес, підвищити корисне навантаження кадрів, розширити адресний простір мережі.

2. Подано корисну модель нового способу передачі даних в E6 мережі із заміщенням мережного та транспортного рівнів універсальною технологією канального рівня, на яку отримано відповідний патент України.

3. Розроблено новий стек протоколів E6, визначено інтерфейси з прикладним та канальним рінями еталонної моделі взаємодії відкритих систем, а також функції проміжного рівня Узгодження E6, якій забезпечує використання стандартних засобів Ethernet LLC1/2 замість протоколів UDP/TCP відповідно.

4. Розроблено принципи реалізації E6 мереж на основі нової схеми доставки пакету (кадру), яка має суттєві переваги перед IPoverEthernet, що полягають у відсутності повторної інкапсуляції пакетів та відображення адрес, а також у можливості прийняття маршрутного рішення по перших 6-ти байтах кадру.

5. Розроблено дистанційно-векторний протокол динамічної маршрутизації E6 мереж E6RIP на основі відповідного протоколу RIP, що дозволяє автоматично заповнювати адресні таблиці пристроїв та забезпечує адаптацію до змінної структури мережі без використання широкомовлення.

2 ОЦІНКА ЕФЕКТИВНОСТІ E6 МЕРЕЖ

Як основу для порівняльної оцінки ефективності адресації E6 використано алгоритми формування й доставки пакетів (кадрів) [25], а також основні архітектурні особливості сучасних мережних пристроїв, що забезпечують доставку в глобальних мережах – маршрутизаторів [1]. Основним сучасним сімейством мережних протоколів є стек TCP/IP [24,31]; основною технологією канального й фізичного рівнів еталонної моделі взаємодії OSI-ISO - технологія Ethernet [33,37]. Тому як еталон для порівнянь обрана інкапсуляція IP повз Ethernet (IPoverEthernet), представлена в стандарті [49], яка отримала свій подальший розвиток у додаткових стандартах [62,67]. У якості еталонних архітектурних рішень сучасних маршрутизаторів використана архітектура, розроблена й реалізована компанією CISCO [1] - одним із провідних виробників мережних пристроїв.

Оцінки базуються на імовірнісному підході до аналізу процесів доставки пакету (кадру) і застосуванні стохастичних методів [5]. Як основні показники ефективності обрано: обсяг корисного навантаження пакетів (payload) і час доставки пакета (delivery time). Збільшення корисного навантаження пакетів спричиняє розширення пропускної здатності мережі при використанні наявних ліній зв'язку й чипсетів мережних пристроїв, а час доставки пакета є одним з основних показників якості обслуговування (QoS) [25], особливо важливих у додатках реального часу, у тому числі телефонії [12].

Хоча застосування E6 адресації припускає також модифікацію алгоритмів гарантованої доставки інформації, пов'язану з використанням можливостей Ethernet LLC2 [51] замість процедур ковзного вікна TCP [65], у дійсному дослідженні оцінюється вплив цього фактора лише на збільшення корисного навантаження. Питання порівняння ефективності протоколів TCP і Ethernet LLC2 не розглядаються в роботі; попередні оцінки дозволяють затверджувати, що при практично рівних функціональних можливостях, Ethernet LLC2 має переваги, обумовлені більше швидкою реакцією на втрату пакетів [25,33].

2.1 Переваги E6 мереж

Попередній аналіз E6 мереж дозволяє зазначити їх наступні переваги:

1 Безсумнівною перевагою схем адресації E6/E6-4/E4P є анулювання додаткової інформації мінімально на 36 байтів для кожного сегмента за рахунок ліквідації заголовків TCP, IP. Крім того, при розбивці сегмента на пакети скорочується 16 байтів заголовків IP для кожного

пакета. При розгляді пакетів з опціями TCP і IP скорочується більший простір. И. хоча анулювання заголовків дозволяє визволити лише близько 3% простору кадру Ethernet максимальної довжини (36/1500), переваги можуть бути істотними при передачі VoIP трафіку [12]. Розміри TCP/IP заголовків порівнянні з довжиною комірки ATM технології [22], розробленої саме виходячи з вимог якісної передачі голосового трафіку. Заголовки TCP/IP є занадто обтяжним вантажем при передачі невеликих пакетів IP-телефонії.

2 Крім того, E6 ліквідує подвійну адресацію в мережі і зв'язані з нею проблеми відображення адрес, що обумовлювали розробку додаткових протоколів, таких як ARP/RARP [31,62]. Відображення адрес на кожному маршрутизаторі вносило непередбачені затримки до процесу доставки IP пакету у разі відсутності відповідного запису у таблиці та необхідності виконання ARP-запитів у мережі.

3 Стосовно стандартної інкапсуляції IPoverEthernet [49] додатковою перевагою E6 є відсутність застосування широкомовлення у процесі доставки кадру, що дозволяє уникнути непродуктивного використання мережі для доставки зайвих пакетів та непередбачених затримок зумовлених широкомовними штормами.

4 Використання схеми адресації E6 дозволяє прийняти маршрутне рішення по перших 6-ти байтах кадру, та не потребує деінкапсуляції пакету на відміну від IP пакетів [24,25,31]. Це дає можливість для застосування швидкого перенаправлення пакету до порту призначення та ретрансляції кадру одночасно із його прийманням [1], що суттєво скорочує час доставки пакету у мережі.

5 Застосування схеми адресації E6 дозволяє розширити адресний простір мережі в 16 тисяч разів і вирішити в такий спосіб проблему дефіциту адрес.

Зазначені переваги E6 мереж вимагають більш точних кількісних порівняльних оцінок, які буде здобуто у наступних підрозділах.

2.2 Оцінка корисного навантаження пакетів

Попередні порівняння виконано методом прямих розрахунків. Більш досконалі порівняльні оцінки мають бути отримані за допомогою розробки спеціальних аналітичних та імітаційних моделей (Розділ 3, 4).

Технологія E6 передбачає анулювання заголовків TCP [65], UDP [63], IP [64] та додання заголовку HP2 (Розділ 1); довжина заголовків наведена в Табл. 2.1. Наявність додаткових опцій протоколів TCP, IP вимагає використання оцінок середньої довжини заголовків.

Оскільки основна корисна інформація мережі передається через протокол TCP, а протокол UDP використовується для передачі службових повідомлень (DNS), розглянемо інкапсуляцію TCP-IP-Ethernet. Стандартна довжина (LHEH) заголовку кадру Ethernet (IEEE 802.3) складає 14 байтів.

Результати оцінок суттєво залежать від довжини пакетів яка зумовлена певним застосуванням прикладного рівня мережі. Розглянемо три типи застосувань: телефонія VoIP [12], гіпертекст WWW [25], мультимедіа Audio/Video [31], – які характеризуються малою, середньою та максимальною довжиною відповідно (Табл. 2.2).

Таблиця 2.1 – Довжина заголовків пакетів

Протокол	Мінімальна довжина (байтів)	Середня довжина (байтів)	Позначення
TCP	20	40	LHTCP
UDP	8	8	LHUDP
IP	20	40	LHIP
E6 (HP2)	4	4	LHE6

Таблиця 2.2 – Довжина корисної інформації пакетів

Застосування	Середня довжина (байтів)	Позначення
VoIP	100	LPVoIP
WWW	500	LPWWW
Audio/Video	1500	LPAV

Заголовок IP присутній в кожному пакеті. Крім того, на оцінки впливає довжина сегменту TCP, від якої залежить наявність в першому пакеті сегменту заголовку TCP. Будемо вважати, що сегмент складається з одного пакету. Тоді загальна оцінка середньої довжини заголовків TCP-IP може бути подано як:

$$LHTCPIP = LHTCP + LHIP = 40 + 40 = 80 \text{ (байтів)}$$

Скорочення довжини заголовків за використанням E6

$$CE6 = LHTCPIP - LHP2 = 80 - 4 = 76 \text{ (байтів)}$$

Тоді доля корисної інформації, яка характеризує ефективність обміну інформацією, може бути оцінено за наступною формулою:

$$F = LP / (LP + LH),$$

де LP – довжина корисної інформації, LH – довжина заголовку. Результати оцінок ефективності обміну інформацією наведено в Табл. 2.3, 2.4.

Таблиця 2.3 – Оцінки довжини інформації

Застосування	TCP/IP		E6	
	Загальна	Корисна	Загальна	Корисна
VoIP	194	100	118	100
WWW	594	500	518	100
Audio/Video	1594	1500	1518	1500

Таблиця 2.4 – Оцінки ефективності обміну інформацією

Застосування	Ефективність TCP/IP (%)	Ефективність E6 (%)
VoIP	51,55	84,75
WWW	84,18	96,53
Audio/Video	94,1	98,81

Отже з Табл. 2.4 випливає висновок, що технологія E6 ефективніша за всіх типів застосувань мережі. Найбільша ефективність, майже вдвічі відповідно до TCP/IP, досягається на застосуваннях телефонії, що може суттєво вплинути також на скорочення часу доставки пакетів, найбільш важливого показника, який зумовлює можливість використання мереж з комутацією пакетів для цього типу сервісу.

2.3 Аналіз алгоритмів доставки пакетів

Розглянемо схеми доставки пакетів між кінцевими вузлами (хостами) мережі для E6 і IP (Рис. 1.5, 1.6). Якщо не розглядати вплив сеансових і транспортних протоколів, алгоритми роботи хостів аналогічні алгоритмам роботи проміжних маршрутизаторів. Алгоритми обробки пакетів у традиційних IP-маршрутизаторах (IPR) і комутуючих маршрутизаторах E6 (KME6)

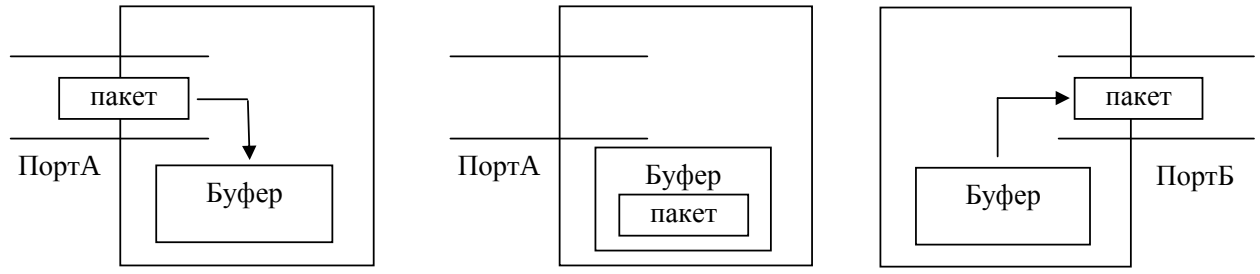
істотно відрізняються, що спричиняє поліпшення характеристик пропускної здатності і якості обслуговування Е6 мереж у порівнянні з IP-мережами.

Аналіз архітектурних особливостей сучасних маршрутизаторів [1,24] дозволяє виділити наступні основні класифікаційні ознаки:

1. Буферування пакета
   1.1. Обов'язкове буферування (store-and-forward)
   1.2. Безпосередня передача між портами (cut-through)
2. Кеш маршрутів
   2.1. Без кешування маршрутної інформації
   2.2. Використання кешу маршрутної інформації
3. Форма подання маршрутної інформації
   3.1. Хеш таблиця
   3.2. 256-дерева

Оскільки сучасні високопродуктивні маршрутизатори в основному реалізують наступний набір можливостей [1]: кешування маршрутної інформації в портах, безпосередня передача пакетів між портами, зберігання маршрутної інформації у формі 256-дерев, - основні порівняння будуть виконані для зазначеного набору ознак; у противному випадку ознаки будуть вказано окремо.

Для порівняння алгоритмів розглянемо більш докладно архітектурні особливості маршрутизаторів, представлені на Рис. 2.1, 2.2.



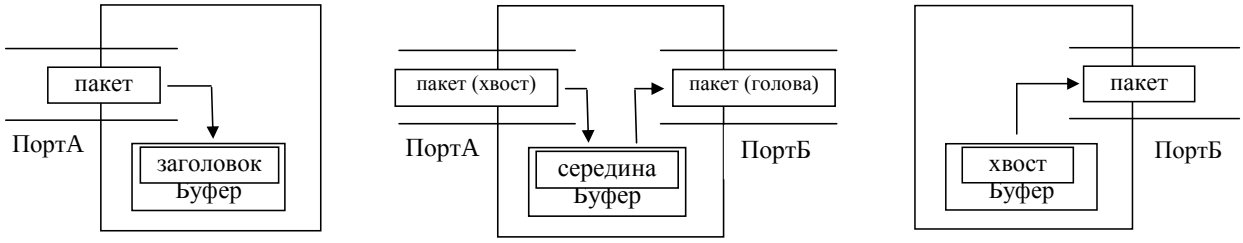
а) прийом пакета до буфера б) обробка пакета в) передача пакета з буфера

Рисунок 2.1 – Схема проходження пакета в маршрутизаторі
з обов'язковим буферуванням пакета

Таким чином, затримка пакета при проходженні маршрутизатора з обов'язковим буферуванням складається із часу прийому пакета до буфера й часу обробки. При цьому час

обробки містить у собі час ухвалення маршрутного рішення (перемикання) і час очікування пакетом звільнення порту.

На Рис. 2.2 представлена схема проходження пакета в маршрутизаторі з безпосередньою передачею між портами. На відміну від повторювача, що ретранслює прийнятий пакет практично без затримок, маршрутизатор повинен ухвалити рішення щодо вибору порту перенаправлення. Для прийняття такого рішення необхідно мати заголовок пакета, принаймні, його частину, що містить адресу призначення. Так, наприклад, кадр Ethernet містить адресу призначення в перших шести октетах. Таким чином, буферується початок кадру, що включає адресу призначення й деяка кількість байтів, що буде отримано за час ухвалення маршрутного рішення. У випадку, якщо порт призначення вільний, кадр починає ретранслюватися одночасно із процесом його прийняття в буфер. Мінімальний розмір буфера дорівнює довжині адреси призначення плюс кількість бітів, що надходять за час ухвалення маршрутного рішення у відповідності зі швидкістю передачі даних у каналі. У випадку, якщо порт призначення зайнятий, використається раніше розглянута схема з буферуванням цілого пакета.



а) прийом заголовка пакета в буфер

б) ретрансляція прийнятого пакета

в) ретрансляція пакета з буфера

Рисунок 2.2 – Схема проходження пакета в маршрутизаторі з безпосередньою передачею пакета між портами

Виберемо позначення для розглянутих часових характеристик відповідно до Табл. 2.5 для подальших розрахунків.

Таблиця 2.5 – Позначення часових параметрів буферування

Позначення	Опис
τPkt	Час прийому пакета
$\tau PktHdr$	Час прийому заголовка пакета
$\tau RtDc$	Час ухвалення маршрутного рішення

Ухвалення маршрутного рішення може виконуватися центральним вузлом і може бути розподіленим по портах. В основі алгоритмів ухвалення маршрутного рішення лежить адресна таблиця маршрутизатора. У розподіленому підході використаються копії адресної таблиці й реплікації основної таблиці по портах. Розмір адресної таблиці може бути значним – десятки тисяч рядків, що істотно збільшує час ухвалення маршрутного рішення.

Для скорочення часу ухвалення маршрутного рішення використається кешування маршрутної інформації в спеціальній швидкій пам'яті порту, іменованому кеш. Організація алгоритмів заповнення кешу заснована на витисненні старих і невикористовуваних записів. Роботи присвячені оцінкам ефективності кешування зазначені в [1,24,25]; результати можуть бути коротко сформульовані в такий спосіб: кеш ефективний у маршрутизаторах близьких до периферії, при передачі довгих послідовностей пакетів (мультимедіа) між фіксованими вузлами мережі. У цьому випадку для кожного наступного пакета з кешу витягається результат маршрутного рішення, збережений для деякого попереднього (першого) пакета. Спочатку порт виконує швидку перевірку в кеші й при відсутності відповідного запису звертається до центрального вузлу маршрутизації, що аналізує повну адресну таблицю. Однак у магістралях ефективність кешу істотно знижується через різнорідну адресну інформацію пакетів і неминуче швидке витиснення запису.

Подальші оцінки часу ухвалення маршрутного рішення $\tau RtDc$ будуть виконані на основі характеристик наведених у Табл. 2.6.

Таблиця 2.6 – Позначення параметрів маршрутного рішення

Позначення	Опис
$\tau Cash$	Час звертання до кэшу
τRt	Час звертання до повної адресної таблиці
$pCash$	Імовірність наявності запису в кэше

При роботі з маршрутною інформацією істотним фактором, що впливає на час ухвалення маршрутного рішення, є форма подання маршрутної інформації. Найбільш простою є організація маршрутної інформації у формі неупорядкованої таблиці, тоді час пошуку пропорційний половині довжини таблиці. Упорядковані таблиці й хешування є більше ефективним способом, що застосовувався в деяких ранніх моделях маршрутизаторів CISCO [1].

Практично всі сучасні маршрутизатори використають для подання маршрутної інформації 256-дерева [1]. У деяких випадках кеш може бути організований на основі асоціативної пам'яті з паралельним пошуком по всіх записах.

Розглянемо 256-дерева як основну форму подання маршрутної інформації (Рис. 2.3). Адреса розбивається на октети; кожний з послідовно занумерованих октетів задає відповідний рівень деревоподібної структури. У такому випадку час пошуку запису фіксований й пропорційний кількості рівнів дерева - кількості октетів адреси. Дерево може бути не повним через те, що деякі значення октетів відсутні у фактичній маршрутній інформації, а також при використанні не всіх октетів за наявністю певною маски. Надалі викладені оцінки часу пошуку маршруту будуть виконані для подання інформації у формі 256-дерев.

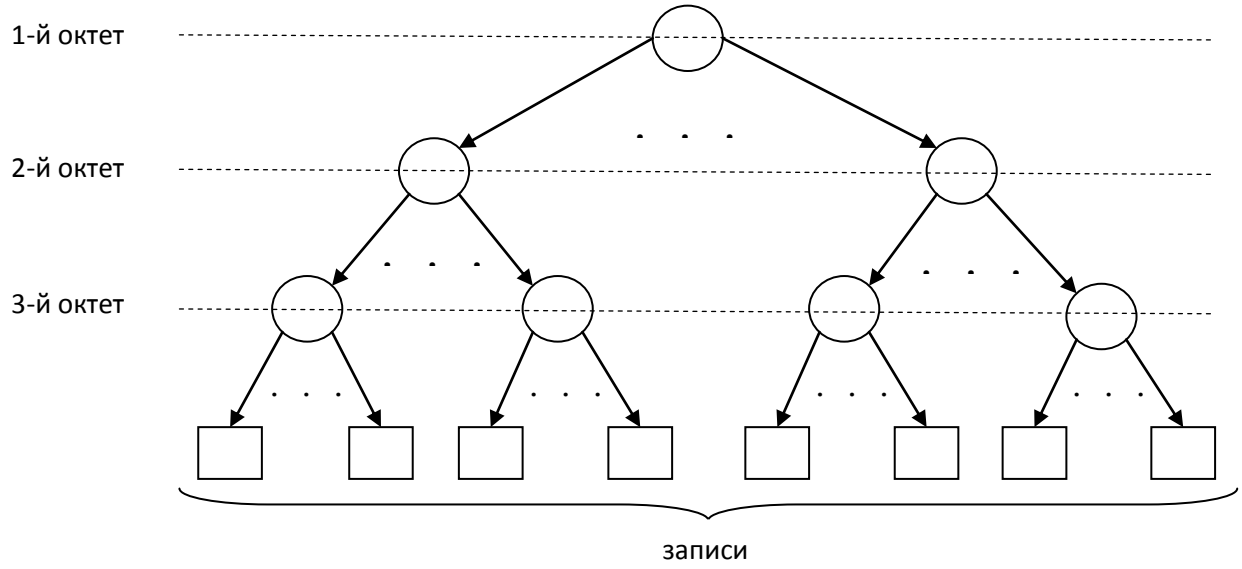


Рисунок 2.3 – Приклад 256-дерева (три рівні)

Для подальшого порівняльного аналізу представимо алгоритми (Рис. 2.4, 2.5) проходження пакетів (кадрів) для маршрутизаторів працюючих за технологіями:

- IPoverEthernet (IPoE);
- E6.

Виділимо для порівняння основні 4 варіанти (Табл. 2.7), які виходять у результаті комбінації виділених архітектурних особливостей сучасних маршрутизаторів.

Таблиця 2.7 – Позначення типів архитектур і мережних технологій

Архітектура		Технологія	
		IPoverEthernet	E6
Тип архітектури	Store-and-forward	sfIPoE	sfE6
	Cut-through	ctIPoE	ctE6

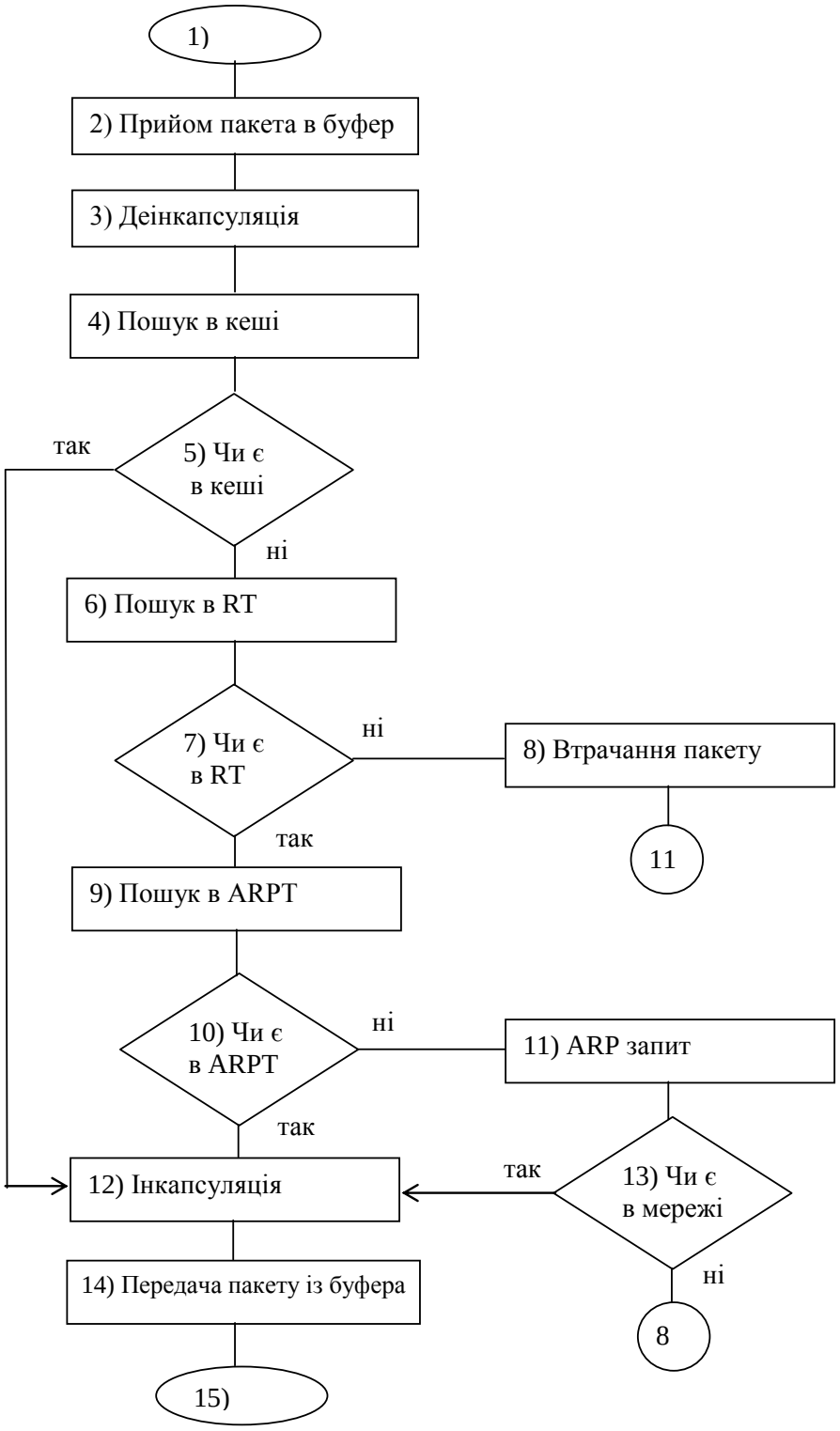


Рисунок 2.4 – Алгоритми обробки (перемикання) пакета в технології IPoverEthernet (sfIPoE)

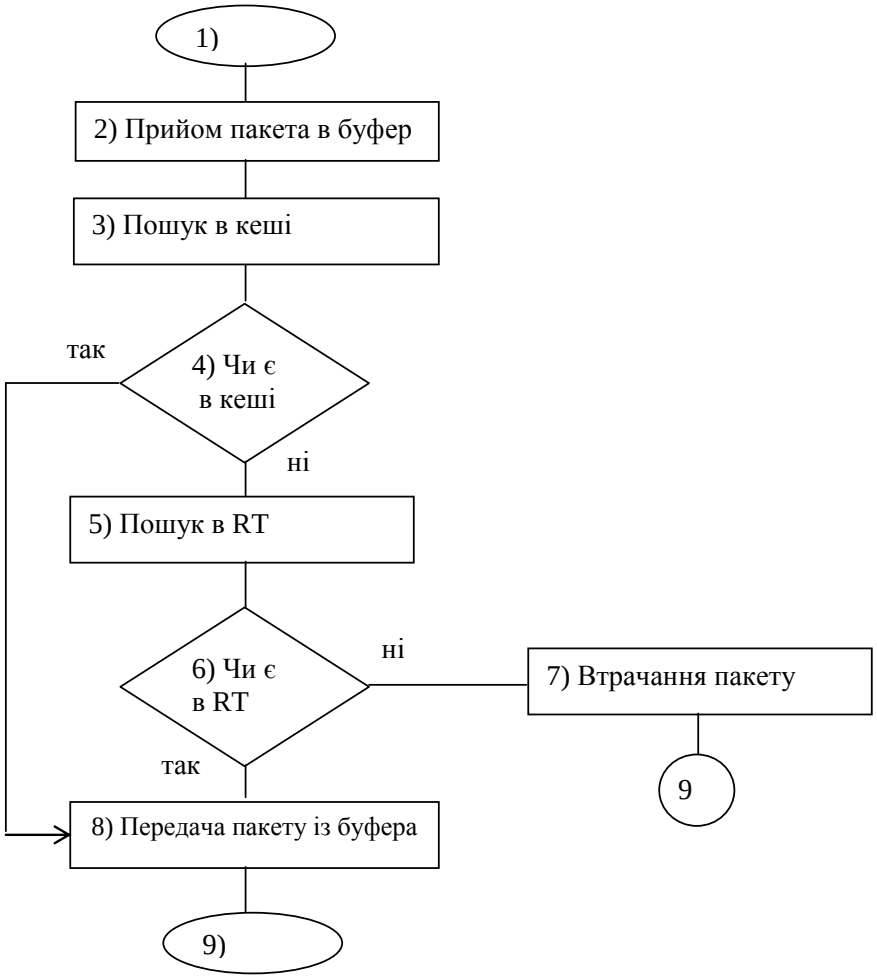


Рисунок 2.5 – Алгоритми обробки (перемикання) пакета в технології E6 (sfE6)

Основна відмінність ct алгоритмів полягає в тому, що на кроці 2 у буфер приймається тільки початкова частина пакета. Прийом до буфера частини пакета, що залишилася, виконується паралельно з іншими кроками алгоритмів. При влученні в заключний блок (14 - IPo, 8 - E6) передача пакета з буфера виконується паралельно з його прийомом.

Дві технології істотно відрізняються довжиною початкової частини пакета (кадру), необхідної для запуску алгоритму обробки пакета, опис якої наведено в Табл. 2.8.

Таблиця 2.8 – Стартова частина пакета (кадру)

Технологія	Заголовки		Довжина (октетів)
IPoE	EH	IPH	14+20=34
E6	EH (DA)		6

Де EH - заголовок Ethernet, IPH - заголовок IP, DA - поле адреси призначення заголовка EH.

Таким чином, з аналізу Табл. 2.8 можна зробити висновок про певні переваги технології E6, пов'язаних зі скороченням стартової затримки в 5,6 разів у порівнянні з IPoE, що дозволяє істотно скоротити час доставки пакета при використанні Cut-through архітектури. Зазначимо, що саме розміщення позначки у голові пакету в значній мірі зумовило можливості технології MPLS [68,69] по прискоренню доставки пакетів та її розповсюдження в сучасних магістралях операторів зв'язку [3].

2.4 Оцінка часу доставки пакетів

На оцінку часу доставки пакета істотно впливає фактор наявності черг у мережних пристроях (комутаторах, маршрутизаторах) [5,19,21]. Первісні оцінки виконаємо в припущенні відсутності черг; у цьому випадку досягається мінімально можливий час доставки пакета. Відмінності часу доставки при відсутності черг пов'язані з наявністю або відсутністю маршрутного запису в кеші, що спричиняє необхідність використання мінімальних, максимальних і середніх значень.

Оцінимо час обробки пакета мережним пристроєм:

$$
\begin{aligned}
&\tau 1IPoE_{\min} = 2\cdot \tau Enc + \tau CashIP,\\
&\tau 1IPoE_{\max} = 2\cdot \tau Enc + \tau CashIP + \tau RtIP + \tau ARP + \tau AR\,\mathrm{Pr}\,eq,\\
&\tau 1IPoE_{avr} = 2\cdot \tau Enc + \tau CashIP + (1-pCash)\cdot(\tau RtIP + \tau ARP + (1-pARP)\cdot \tau AR\,\mathrm{Pr}\,eq),\\
&\tau AR\mathrm{Pr}eq = 2\cdot \tau Seg + 2\cdot \tau Enc,\\
&\tau 1E6_{\min} = \tau CashE6,\\
&\tau 1E6_{\max} = \tau CashE6 + \tau RtE6,\\
&\tau 1E6_{avr} = \tau CashE6 + (1-pCash)\cdot \tau RtE6.
\end{aligned}
\qquad (2.1)
$$

Необхідною умовою для початку обробки пакета в sf маршрутизаторах є буферування цілого пакета, що виконується за час τSeg; для маршрутизаторів ct – буферування адресної частини: $\tau SegDA$ і $\tau SegIPE$. Найбільш прості оцінки одержуємо для sf маршрутизаторів; у цьому випадку час обробки пакета маршрутизатором підсумується із часом передачі в сегменті й множиться на загальне число сегментів. Кількість проміжних мережних пристроїв позначимо як qR. Тоді час доставки може бути оцінене як:

$$\tau sfIPoE = \tau Seg + (\tau Seg + \tau 1IPoE) \cdot qR,$$
$$\tau sfE6 = \tau Seg + (\tau Seg + \tau 1E6) \cdot qR. \quad (2.2)$$

Для маршрутизаторів ct оцінки трохи видозмінюються: час передачі пакета в сегменті враховується один раз, а потім підсумуються лише часи обробки на маршрутизаторах:

$$\tau ctIPoE = \tau Seg + (\tau SegIPE + \tau 1IPoE) \cdot qR,$$
$$\tau ctE6 = \tau Seg + (\tau SegDA + \tau 1E6) \cdot qR. \quad (2.3)$$

На основі побудованих формул (2.1), (2.2) можуть бути виконано оцінки часів доставки пакетів для обраних технологій і типів маршрутизаторів. Однак, найбільш складною частиною розрахунків є одержання вихідної інформації, пов'язаної з характеристиками виконання окремих етапів обробки пакетів у маршрутизаторах. Як правило, виробник надає інформацію або про максимальну продуктивність маршрутизатора в пакетах у секунду [1,24], що досягається при повністю збалансованій передачі пакетів мінімальної довжини між парами портів, або про продуктивність внутрішньої шини маршрутизатора [1,31].

Для одержання відносних оцінок у більше простій формі використаємо додаткові допущення. Нехай продуктивність сегмента дорівнює BR бітів у секунду, а продуктивність мережного пристрою в k раз більше й дорівнює $k \cdot BR$. Приймемо за одиницю кількість операцій Nop, які необхідно виконати при маршрутизації одного пакета в технології IPoE. Використаємо експертні оцінки відносної трудомісткості окремих етапів обробки пакета, представлені в Табл. 2.9.

Таблиця 2.9 – Оцінки трудомісткості виконання операцій

Етап	*Enc*	*CashIP*	*RtIP*	*ARP*
Трудомісткість	0,1	0,1	0,5	0,2
Позначення	k_{Enc}	k_{CashIP}	k_{RtIP}	k_{ARP}

При зберіганні інформації в формі 256-дерева час виконання операції пропорційний кількості рівнів дерева, тому при порівнянні часів виконання відповідних операцій IPoE і E6 технологій будемо використати наступний коефіцієнт: $k_{E6} = 6/4 = 1{,}5$. Тоді з (2.1) одержимо:

$$\tau 1IPoE_{\min} = \frac{2 \cdot Nop \cdot k_{Enc} + Nop \cdot k_{CashIP}}{k \cdot BR} = \frac{Nop}{k \cdot BR} \cdot \left(2 \cdot k_{Enc} + k_{CashIP}\right),$$

$$\tau 1E6_{\min} = \frac{Nop}{k \cdot BR} \cdot k_{CashIP} \cdot k_{E6},$$

$$\tau 1IPoE_{\max} = \frac{Nop}{k \cdot BR} \cdot \left(4 \cdot k_{Enc} + k_{CashIP} + k_{RtIP} + k_{ARP}\right) + 2 \cdot \frac{Lpkt}{BR},$$

$$\tau 1E6_{\max} = \frac{Nop}{k \cdot BR} \cdot \left(k_{CashIP} + k_{RtIP}\right) \cdot k_{E6}, \qquad (2.4)$$

$$\tau 1IPoE_{avr} = \frac{Nop}{k \cdot BR} \cdot \left(2 \cdot k_{Enc} + k_{CashIP} + \left(1 - pCash\right) \cdot \left(k_{RtIP} + k_{ARP}\right)\right) +$$

$$\left(1 - pCash\right) \cdot \left(1 - pARP\right) \cdot \left(\frac{Nop}{k \cdot BR} \cdot 2 \cdot k_{Enc} + 2 \cdot \frac{Lpkt}{BR}\right),$$

$$\tau 1E6_{avr} = \frac{Nop}{k \cdot BR} \cdot \left(k_{CashIP} + \left(1 - pCash\right) \cdot k_{RtIP}\right) \cdot k_{E6}.$$

Для розрахунку загального часу доставки підставимо (2.4) в (2.2), (2.3) і одержимо наступні формули:

$$\tau sfIPoE = \frac{Lpkt}{BR} + \left(\frac{Lpkt}{BR} + \tau 1IPoE\right) \cdot qR,$$

$$\tau sfE6 = \frac{Lpkt}{BR} + \left(\frac{Lpkt}{BR} + \tau 1E6\right) \cdot qR,$$

$$\tau ctIPoE = \frac{Lpkt}{BR} + \left(\frac{Lipe}{BR} + \tau 1IPoE\right) \cdot qR, \qquad (2.5)$$

$$\tau ctE6 = \frac{Lpkt}{BR} + \left(\frac{Lda}{BR} + \tau 1IPoE\right) \cdot qR.$$

Для порівняльної оцінки варто знайти відносини часів доставки пакета (2.5) технологій E6 і IPoE; позначимо відповідні відносини як $\rho = {}^{\tau_{E6}}/_{\tau_{IPoE}}$.

2.5 Порівняльна оцінки якості обслуговування E6 та IP мереж

Традиційно для оцінки продуктивності і якості обслуговування телекомунікаційних мереж [5] застосовують Марковські процеси й сіті масового обслуговування [9,19,21]. Основним параметром використовуваних моделей є час обробки пакета на пристрої - час

обслуговування, або деякий закон його розподілу. Однак, відомі моделі абстрагуються від алгоритмів обробки пакета й архітектурних особливостей пристроїв [1]. Оскільки основні переваги технології E6 досягаються на архітектурах з безпосередньою передачею пакета між портами, необхідно використовувати оцінки часу обслуговування, що враховують одночасність процесів прийому пакета до вихідного порту і його ретрансляції в порт призначення.

Порівняльна оцінка технологій E6 і IPoE виконана за допомогою отриманих співвідношень (2.5). При значенні параметрів, вказаних у Табл. 2.10, за допомогою програми Excel отримані оцінки часу доставки пакета, наведені в Табл. 2.11.

Таблиця 2.10 – Обрані значення параметрів

kEnc	kCashIP	kRtIP	kARP	k6	pCash	pARP	k	Lpkt	BR	q	Nop	Lipe	Lda
0,1	0,1	0,5	0,2	1,5	0,6	0,6	10	12208	10^9	10	10^3	272	48

Таблиця 2.11 – Оцінки часу доставки пакета

	$\tau 1IPoE$	$\tau 1E6$	ρ	$\tau sfIPoE$	$\tau sfE6$	ρ	$\tau ctIPoE$	$\tau ctE6$	ρ
Max	24,5 мкс	90 нс	273	379,7 мкс	135,2 мкс	2,8	260,3 мкс	13,6 мкс	19,2
Min	30 нс	15 нс	2	134,6 мкс	134,4 мкс	1,001	15,23 мкс	12,84 мкс	1,19
Avr	3,97 мкс	45 нс	88	174 мкс	134,8 мкс	1,3	54,61 мкс	13,14 мкс	4,16

Таким чином, технологія E6 має найбільші переваги в порівнянні з IPoE у застосуваннях реального часу із жорсткими часовими обмеженнями. Оцінка максимального часу обробки кадру пристроєм в 273 рази менше, що обумовлено відсутністю ARP запитів у мережі під час доставки кадру; у результаті загальний час доставки кадру зменшується в 19 разів.

Наступної по значимості областю застосування є телефонія й телеконференції, де основним обмеженням на можливість надання послуг є середній час доставки. У цьому випадку одержуємо виграш в 88 разів на пристрої й в 4 рази для загального часу доставки.

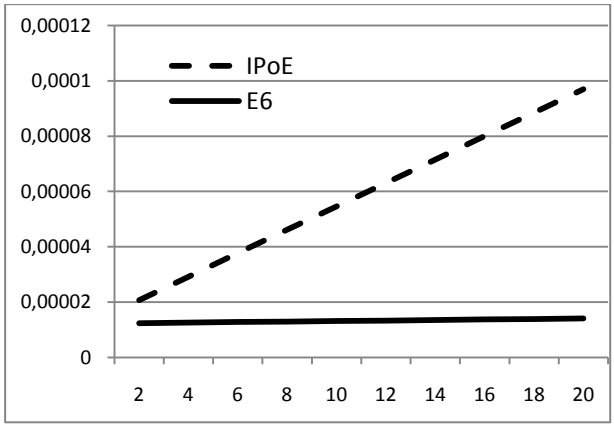
Відзначимо, що найбільші переваги технології E6 можуть бути одержання у випадку використання пристроїв з типом архітектури cut-through, що зумовлює необхідність апаратної реалізації КМЕ6.

На графіках, зображених на Рис. 2.6, представлений вплив основних параметрів на середній час доставки пакета для архітектури cut-through; значення інших параметрів (крім

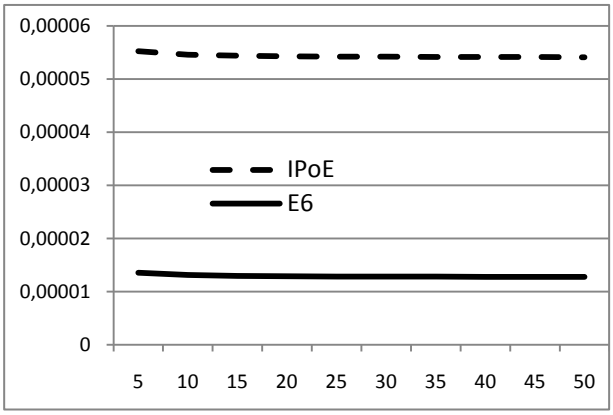
зазначеного на горизонтальній осі) збережені відповідно до Табл. 2.11 ($\rho = 4{,}16$). Аналіз графіків дозволяє зробити висновок, що переваги технології E6 підсилюються при:

– збільшенні довжини маршруту – $\rho = 6{,}9$ на 20 хопах,
– збільшенні продуктивності пристрою – $\rho = 4{,}24$ при k$= 50$,
– зменшенні ефективності кешу – $\rho = 7{,}7$ при $pCash = 0{,}1$ (що відповідає розташуванню в магістралях).

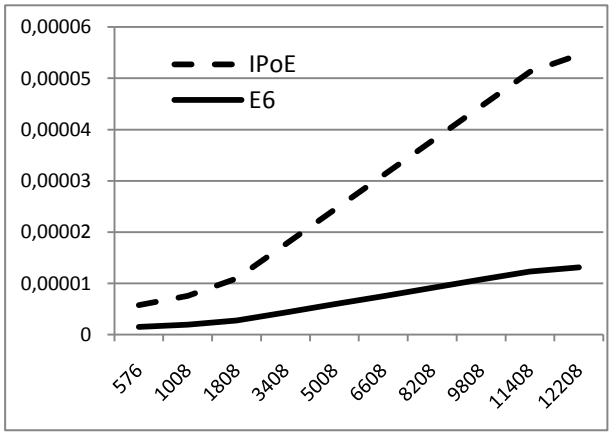
Відзначимо, що зменшення довжини пакета приводить до незначного погіршення показників – $\rho = 3{,}82$ на пакетах мінімальної довжини.



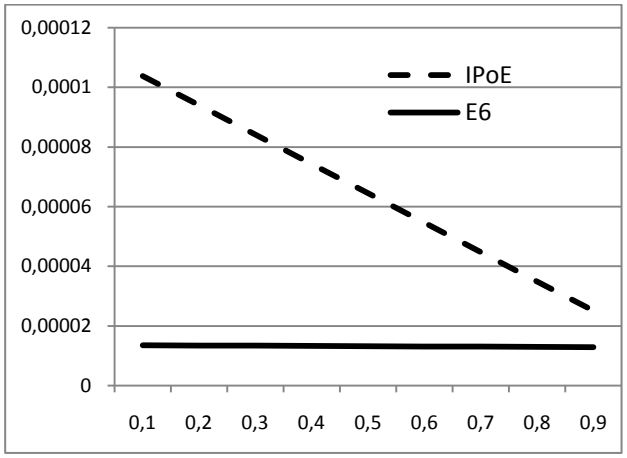
а) від довжини маршруту (у хопах)



б) від продуктивності пристрою (k)



в) від довжини пакета (біт)



г) від ефективності кешу (pCash)

Рисунок 2.6 – Залежність середнього часу доставки пакета (с) від основних параметрів

2.6 Оцінка впливу черг

Наявність черг пакетів у мережному пристрої збільшує час обробки пакета. Для грубої оцінки впливу черг запропоновано використати наступне співвідношення:

$$\tau 1^{Total} = \tau 1^{\text{Proc}} + \tau 1^{Queue}, \qquad (2.6)$$

де $\tau 1^{Total}$ – загальний час обробки, $\tau 1^{\text{Proc}}$ – час виконання операцій обробки (3.4), $\tau 1^{Queue}$ – час перебування в черзі.

Для оцінки часу перебування пакета в черзі до порту призначення $\tau 1^{Queue}$ запропоновано використати Марковські процеси з дискретним часом [5,9]. Як такт часу Δt виберемо час обробки пакета у випадку відсутності черг. Основними параметрами ланцюга Маркова, що моделює роботу вихідного каналу деякого порту, є: n – загальна кількість портів пристрою; q – імовірність надходження пакета із вхідного каналу деякого порту пристрою. Стан $s_i, i \geq 0$ у цьому випадку повністю характеризується кількістю пакетів i; при цьому пакет перебуває в черзі в станах $s_i, i > 1$.

Приклад ланцюга Маркова при $n = 4$ представлений на Рис. 2.7. Відзначимо, що надходження пакетів з кожного порту розглядається незалежно й неможливо надходження більше одного пакета з порту за такт часу; тому число пакетів, що надійшли за такт часу не перевищує $n-1$ (перенапрямок пакета в порт прибуття виключається).

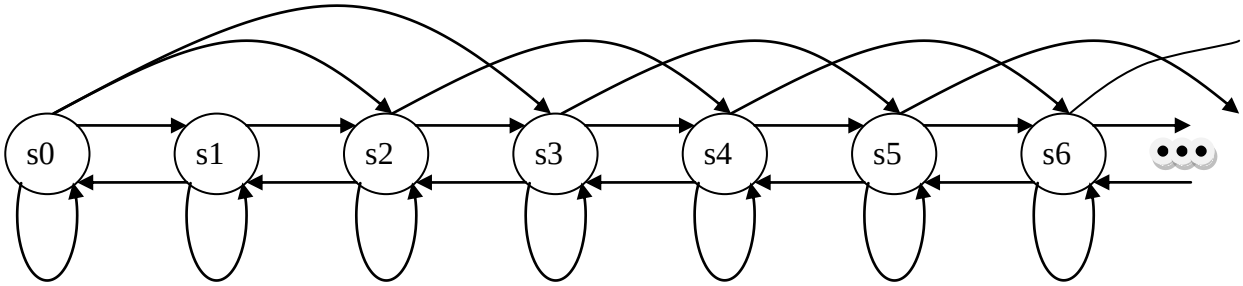


Рисунок 2.7 – Приклад ланцюга Маркова вихідного каналу порту (при $n = 4$)

Марковський ланцюг є нескінченним у випадку відсутності обмежень на розмір буфера пакетів і може бути представлений в параметричній формі (Табл. 2.12). У графі примітки не зазначена подія завершення обробки поточного пакета, що має місце у всіх станах крім s_0.

Обчислення ймовірності наявності черги $p_{Queue} = 1 - \pi_0 - \pi_1$ (де π_i – стаціонарна ймовірність знаходження в стані s_i) дозволяє виконати подальшу оцінку середнього часу перебування пакета в черзі [9,19,21].

Таблиця 2.12 – Параметричне подання ланцюга Маркова

Вихідний стан	Наступний стан	Імовірність переходу	Подія
s_0	s_0	$(1-q)^{n-1}$	Відсутня
s_0	$s_j, j=\overline{1,n-1}$	$q^j \cdot (1-q)^{n-j-1}$	Надійшло j пакетів
$s_i, i>0$	s_{i-1}	$(1-q)^{n-1}$	Відсутня
$s_i, i>0$	$s_{i+j-1}, j=\overline{1,n-1}$	$q^j \cdot (1-q)^{n-j-1}$	Надійшло j пакетів

Висновки до 2 розділу

1. На основі аналізу особливостей формування та передавання пакетів в мережі визначені основні переваги системи адресації E6 перед IP адресацією серед яких є збільшення адресного простору, скорочення заголовків, прискорення алгоритмів обробки пакету на пристрої.

2. За допомогою розрахунків корисного навантаження пакетів зазначено, що система адресації E6 забезпечує більшу продуктивність передавання корисної інформації у мережі відносно до IP-адресації; найкращі результати здобуто для невеликих пакетів телефонії де продуктивність є більшою вдвічі.

3. На основі аналізу архітектур сучасних телекомунікаційних пристроїв та їх алгоритмів обробки пакетів отримано вирази для оцінки часу обробки E6 та IP пакетів на пристрої та часу доставки пакету у мережі, який є одним з найважливіших показників якості обслуговування.

4. Для оцінки впливу черг, які виникають на мережних пристроях побудовано модель у вигляді ланцюга Маркова. Для опису ланцюга з нескінченною множиною станів застосована параметрична форма подання.

5. Отримані порівняльні оцінки якості обслуговування E6 та IP мереж, свідчать про суттєві переваги системи адресації E6 особливо для застосувань реального часу з жорсткими часовими обмеженнями, через те що максимальний час доставки E6 пакетів є меншим в 19 разів. Середній час доставки E6 пакету є меншим в 4 рази, що свідчить про певні переваги E6 для застосувань телефонії та телеконференцій.

6. Досліджено залежність отриманих оцінок від основних параметрів, та зазначено, що переваги E6 посилюються при зростанні довжини маршруту у мережі, зменшенні ефективності кешу (у магістралях), підвищенні продуктивності пристроїв.

3 МОДЕЛЮВАННЯ E6 МЕРЕЖ

Розробка моделей E6 мереж має мету не тільки перевірку працездатності нової системи адресації та вдосконалення оцінок ефективності здобутих у попередньому розділі. Модель являє собою форму специфікації більш точну ніж вербальна [16,32,34]. Отримані у поточному розділі моделі комутуючого маршрутизатору E6 та протоколу динамічної маршрутизації E6-RIP можуть розглядатися як специфікації алгоритмів роботи відповідних пристроїв та програм стеку E6. Ці моделі можуть бути застосовані як початкові у керованої моделлю розробці пристроїв та програмного забезпечення E6 [10,16,46]. Для подання моделей використано розфарбовані сіті Петрі [20,23,26,29,58].

Суттєво істотним для існування глобальної мережі є динамічна маршрутизація [25], яка являє собою адаптацію роботи мережних пристроїв до змінної структури мережі зумовленої як тимчасовим підключенням та відключенням певних пристроїв так і розвитком мережі. Саме тому основну частину досліджень поточного розділу сконцентровано на моделюванні E6 мереж із динамічною маршрутизацією.

3.1 Розробка моделей E6 мереж

Основними компонентами для розробки моделей E6 мереж є комутуючі маршрутизатори E6, моделі протоколу динамічної маршрутизації E6-RIP, моделі термінальних (абонентських) мереж.

3.1.1 Модель комутуючого маршрутизатора KME6

Як базовий компонент обрана модель IP-маршрутизатора [14], яка модифікована з урахуванням використання E6 адрес [45]; виділений компонент, що моделює роботу окремого порту, а також використані рекурсивні функції [73] для роботи з адресною таблицею, представленої списковою структурою.

Модель KME6 збирається шляхом клонування необхідної кількості портів port, доповнених компонентами RIPprocess, RIPupdate і позиціями, що моделюють об'єкти внутрішньої пам'яті. Описи основних типів даних, приведені на Рис. 3.1. Приклад моделі для 4-х портів представлений на Рис. 3.2.

```
colset e6=product INT*INT*INT*INT*INT*INT;
colset mask=INT;
colset nwt=product e6 * mask timed;
colset b=INT timed;
colset pkt=record e6src:e6 * e6dst:e6 * data:b * dt:INT;
colset operation=with REQUEST | RESPONSE;
colset rtm=record opr:operation*nw:nwt *m:INT;
colset cha=union pk:pkt+rm:rtm timed;
colset rtmin=record opr:operation*nw:nwt *m:INT*ifn:INT;
colset rtr=record nw:nwt *m:INT*ifn:INT*chg:BOOL*ta:INT*gcta:INT;
colset buf=product pkt*INT timed;
colset brtm=product rtm*INT;
colset brtr=product rtr*INT;
```

Рисунок 3.1 – Опис основних типів даних (множин кольорів)

Позиції in* задають номери відповідних портів. Позиції p*IN, p*OUT моделюють вхідні і вихідні канали портів відповідно. Внутрішня пам'ять КМЕ6 представлена наступними позиціями: RT – таблиця маршрутизації, Buf – буфер кадрів (пакетів), msg-in – вхідні повідомлення E6-RIP, Bufrt – буфер вихідних повідомлень E6-RIP. Таблиця маршрутизації RT моделюється контактною позицією (I/O) для відображення на головній сторінці моделі.

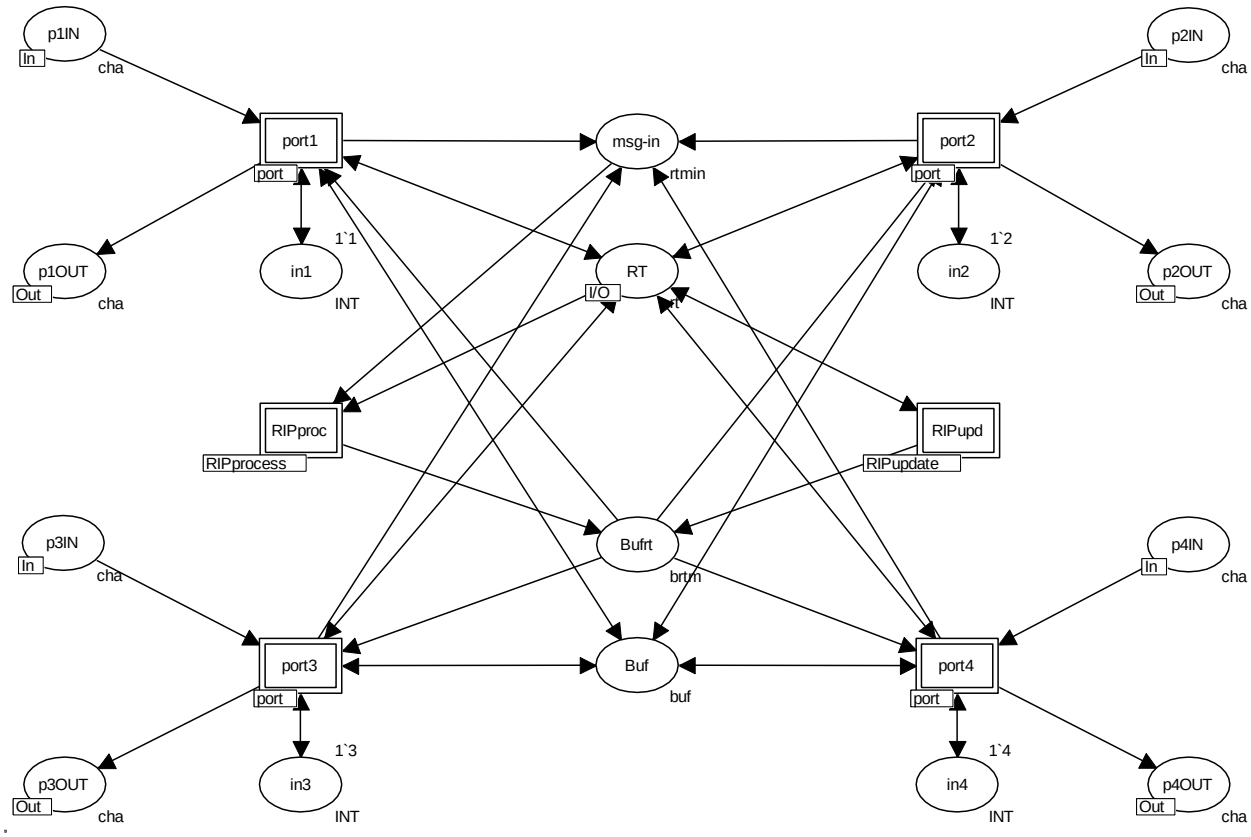


Рисунок 3.2 – Модель КМЕ6 (4 порти)

Модель порту КМЕ6 (port) представлена на Рис. 3.3. Нормальна робота порту забезпечується парою переходів getpkt, putpkt, що виконують прийом вхідного пакета з розміщенням у буфері Buf і передачу пакетів із буфера відповідно. При розміщенні пакета в буфері по таблиці маршрутизації RT визначається порт для перенаправлення пакета. Рекурсивна функція grec знаходить придатний запис таблиці; функція sameNW виконує порівняння Е6 адрес таблиці й адреси призначення пакета (e6dst); записи таблиці з метрикою INFINITY ігноруються. Перехід droppkt моделює втрату пакетів при відсутності адресної інформації; при цьому збільшується лічильник ndrop.

Пари переходів getrtr, putrtr моделюють одержання і відправлення повідомлень протоколу Е6-RIP відповідно. Виконується підрахування отриманих повідомлень Е6-RIP за допомогою лічильника nrtm. Для розрізнення інформації каналу використаний тип даних об'єднання union; ознака pk виділяє звичайні пакети, ознака rm – повідомлення Е6-RIP.

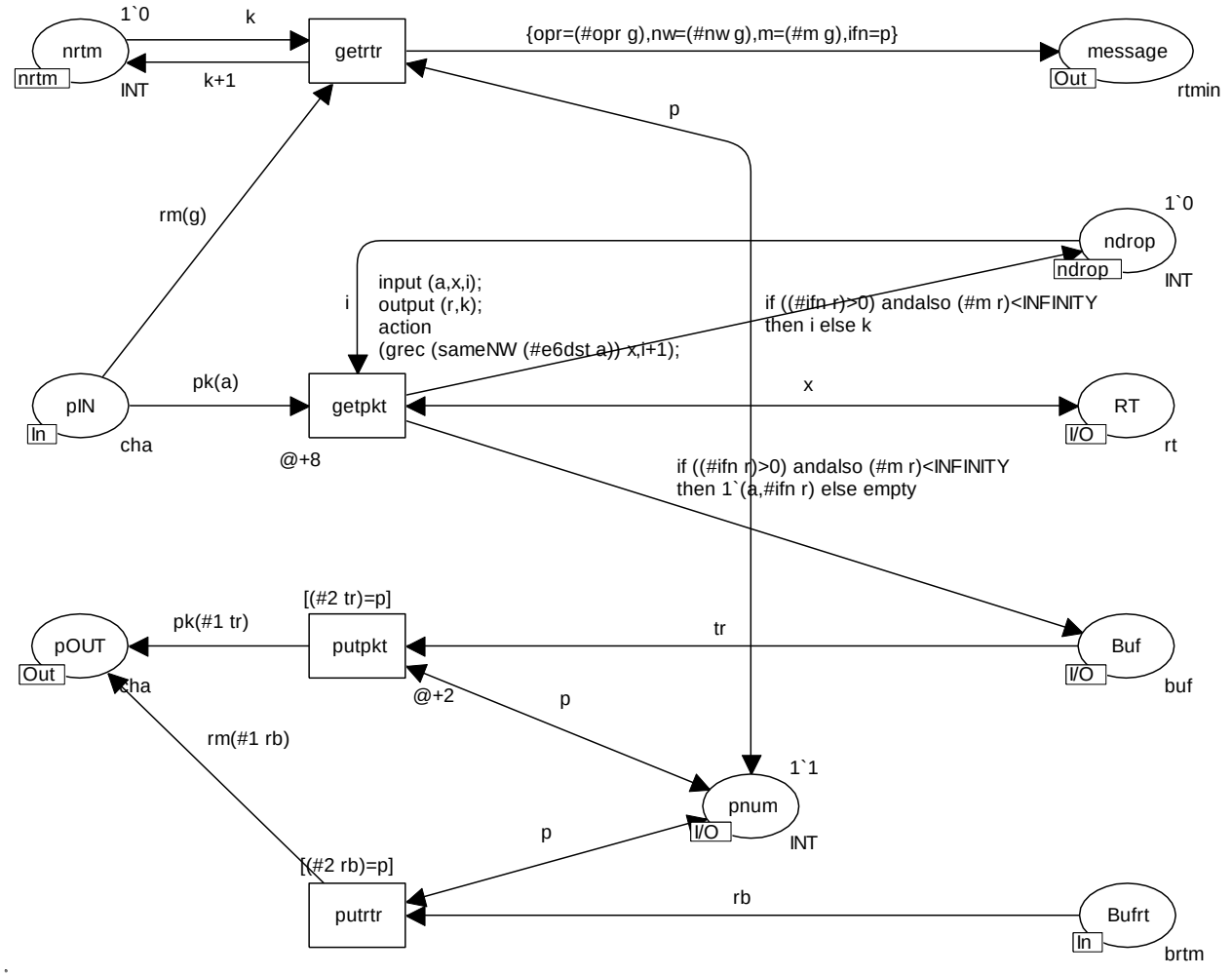


Рисунок 3.3 – Модель порту КМЕ6 (port)

Зупинимося на описах основних типів даних моделі (Рис. 3.1). E6 адреса описана кортежем e6; тип nwt описує адресу E6 мережі/хоста як кортеж, що складається з E6 адреси e6 і маски mask. Пакет pkt складається з E6 адрес відправника e6src і призначення e6dst, умісту data, часового штампа dt для подальшої оцінки якості обслуговування. Формати повідомлення rtm протоколу E6-RIP і записи таблиці маршрутизації rtr відповідають описам підрозділу 1.6. Тип cha описує інформацію каналу, що може бути або пакетом даних pk, або повідомленням E6-RIP rm; у більш складних варіантах можуть використовуватися також ознаки приступності каналу [13,15]. Тип rt описує таблицю маршрутизації як список записів rtr. Типи buf, brtr описують записи вихідних буферів пакетів і маршрутної інформації відповідно; друге поле задає номер порту призначення. Тип rtmin використаний для внутрішнього збереження отриманого повідомлення типу rtm разом з номером вхідного порту ifn.

3.1.2 Компоненти E6-RIP

Протокол E6-RIP (підрозділ 1.6) представлений двома компонентами RIPprocess, RIPupdate, зображеними на Рис. 3.4, 3.5 відповідно. Компонент RIPprocess обробляє отримані повідомлення протоколу E6-RIP, компонент RIPupdate виконує формування відновлень і обробку таймаутів.

Повідомлення E6-RIP надходять у позицію msg-in. Пара переходів resp, req розпізнає команди відповіді і запиту відповідно. При обробці запиту істотно відрізняються запит на передачу всієї таблиці wholeRT і запит про окрему мережу (запис) rec. При формуванні відповіді перехід rec виконує пошук зазначеної в запиті мережі за допомогою функції grec і повертає відповідний запис; при відсутності запису вказується метрика INFINITY. Для передачі всієї маршрутної таблиці вона дублюється в позиції RT1 і передається по записах у порт запиту переходом putrsp; перехід clean виконує очищення порожньої таблиці і повертає ознаку готовності ready. Константа dnw відповідає маршруту по-умовчанню.

При обробці відповіді (маршрутного відновлення) переходом resp повідомлення перетворюється у формат запису таблиці маршрутизації зі збільшенням метрики на одиницю, установкою ознаки зміни chg і таймаута старіння ta; запис зберігається в позиції route. Обробка маршрутного відновлення реалізована функцією newroute в атрибутах переходу process; описи функції newroute і вкладеної функції відновлення запису таблиці procr приведені на Рис. 3.6.

Робота компоненти RIPupdate (Рис. 3.5) ініціюється двома таймерами: регулярних відновлень clock і тригерних відновлень clocki. У першому випадку ретранслюється вся таблиця переходом putr, у другому – один запис (із встановленою ознаки chg) переходом trigger. У

позиції RTp зберігається копія таблиці для наступної ретрансляції по записах переходом putr; перехід clean очищає порожню таблицю. Функція mulrec служить для розмноження запису по всіх портах. Відновлення попадають у позицію route відкіля передаються в буфер переходом put у форматі повідомлення rtm; розщеплення горизонту реалізоване перевіркою на порт джерела (i<>(#ifn r)).

При копіюванні таблиці маршрутизації переходом sieve реалізована її фільтрація відповідно до таймаутів функцією sievert (Рис. 3.6). Додаткову фільтрацію можна виконувати також у переході тригерних змін trigger для більш точного відстеження таймаутів.

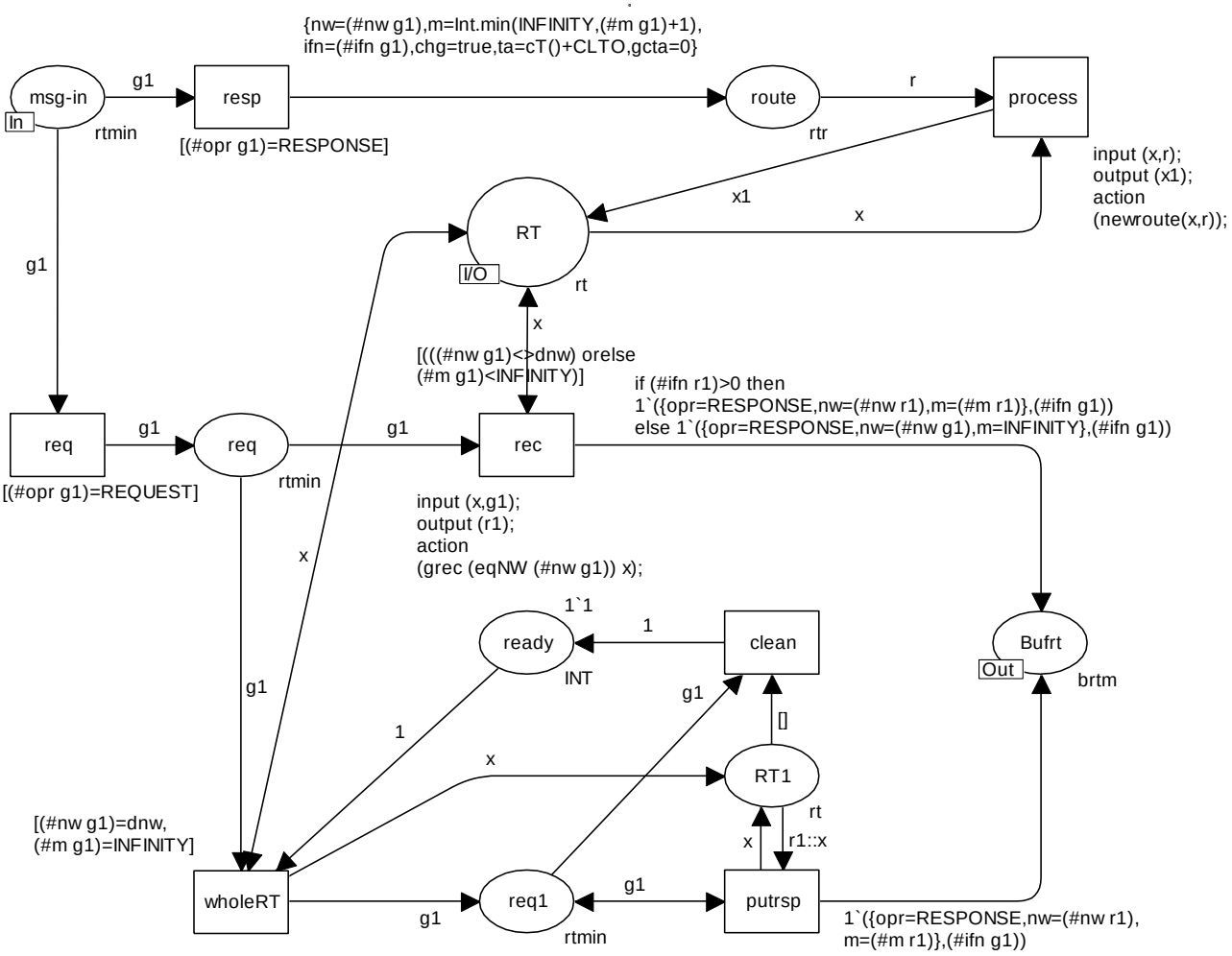


Рисунок 3.4 – Компонент обробки маршрутних повідомлень RIPprocess

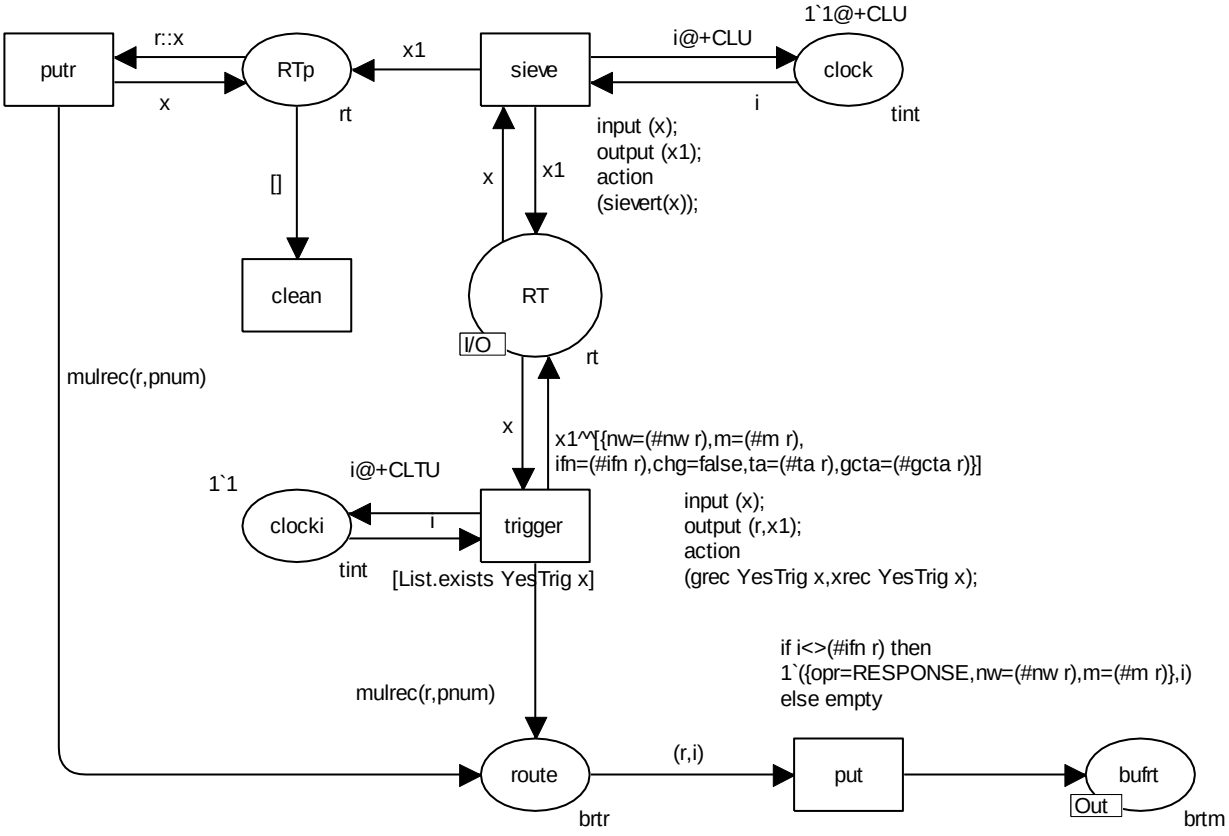


Рисунок 3.5 – Компонент розсилання маршрутних повідомлень RIPupdate

```
fun procr(r0:rtr,r:rtr)=
if (#m r)=INFINITY andalso (#m r0)=INFINITY then r0 else
if (#m r)<INFINITY andalso (#m r0)<INFINITY andalso
   (#ifn r)<>(#ifn r0) andalso (#m r)>=(#m r0) then r0 else
if (#m r)=INFINITY andalso (#m r0)<INFINITY then
   {nw=(#nw r0),ifn=(#ifn r0),m=INFINITY,chg=true,ta=0,gcta=cT()+CLGCTO} else
if (#m r)<INFINITY andalso (#m r)<(#m r0) then r else
if (#m r)<INFINITY andalso(#m r0)<INFINITY andalso
   (#ifn r)=(#ifn r0) andalso (#m r)=(#m r0) then
   {nw=(#nw r0),ifn=(#ifn r0),m=(#m r0),chg=(#chg r0),ta=(#ta r),gcta=0} else r;

fun newroute([],r:rtr)=if (#m r)<INFINITY then [r] else [] |
newroute(r0::t,r:rtr)=if (#nw r0)=(#nw r) then [procr(r0,r)]^^t else [r0]^^newroute(t,r);

fun sievert ([])=[] |
sievert(r::t)=if (#gcta r)>0 andalso (#gcta r)<=cT() then sievert(t) else
  if (#ta r)>0 andalso (#ta r)<=cT() then
    ([{nw=(#nw r),m=INFINITY,ifn=(#ifn r),chg=true,ta=0,gcta=cT()+CLGCTO}]^^sievert(t))
    else ([r]^^sievert(t));

fun mulrec (r,1) = 1`(r,1) | mulrec (r,i)=1`(r,i)++mulrec(r,i-1);
```

Рисунок 3.6 – Функції відновлення маршрутної інформації

3.2 Моделі термінального обладнання з потоковим трафіком

Модель реалістичного трафіка є важливої складовою, що забезпечує загальну адекватність побудованих моделей реальним процесам. Розфарбовані сіті Петрі надають широкі можливості опису як потокового трафіка з різним видом розподілу випадкових величин (рівномірне, Пуасоновське, Ерланга), так і трафіка, що відображає протоколи взаємодії в системах клієнт-сервер [24,25]. У дійсному розділі моделюється потоковий трафік. Термінальна мережа періодично генерує пакети з випадковими адресами джерела і приймача. Причому, адреса джерела знаходиться в діапазону адрес власних мереж, а адреса приймача в діапазоні адрес усіх мереж модельованого фрагмента магістралі Інтернет.

Модель термінальної мережі Terminal, розроблена на основі [14], представлена на Рис. 3.7. Обробка вхідних пакетів представлена простим їх поглинанням за допомогою переходу Counter у верхній лівий частині моделі; при цьому виконується підрахування кількості доставлених пакетів у сполученій позиції Traffic, використовуваної для накопичування статистичної інформації. Крім того, додано дві компоненти traffic та QoS для оцінки трафіка у бітах на секунду та часу доставки пакету відповідно. Означені вимірювальні фрагменти описані у наступному підрозділі.

Генерація пакетів заснована на використанні пари позицій ownNW і allNW, що містять E6-адреси і маски власних мереж і всіх мереж модельованого фрагмента магістралі відповідно. Зауважимо, що контактна позиція ownNW містить адреси у вигляді списку. При ініціалізації моделі запускається перехід gennw, який розбирає список по елементах, та дублює власні підмережі у позиціях ownNW1 та ownNW2. Позицію ownNW1 використано для генерації пакетів даних, а позицію ownNW2 – для генерації пакетів E6-RIP з пропагуванням власних підмереж. Дублювання зумовлено різними періодами генерації відповідних пакетів.

Генерацію пакетів даних виконано у такий спосіб. Адреси відправника та одержувача вибираються в позиції ipsrc і ipdst переходами IpGenerate1, IpGenerate2 відповідно. Найбільш складною дією є генерація випадкової E6-адреси за заданими адресою мережі і маскою. Для цих цілей побудована спеціальна функція gene6(). Циркуляція фішки в послідовності позицій 0, 1, 2 забезпечує циклічне повторення дій. Позиція 0 служить для періодичного запуску генерації пакетів; при цьому часова затримка Delay() на вихідній дузі переходу genPKT задає періодичність генерації. Позиція Data моделює вміст пакетів. Власне формування вихідного пакета виконується переходом genPKT шляхом об'єднання адрес джерела і приймача, а також даних; пакет надходить до позиції out.

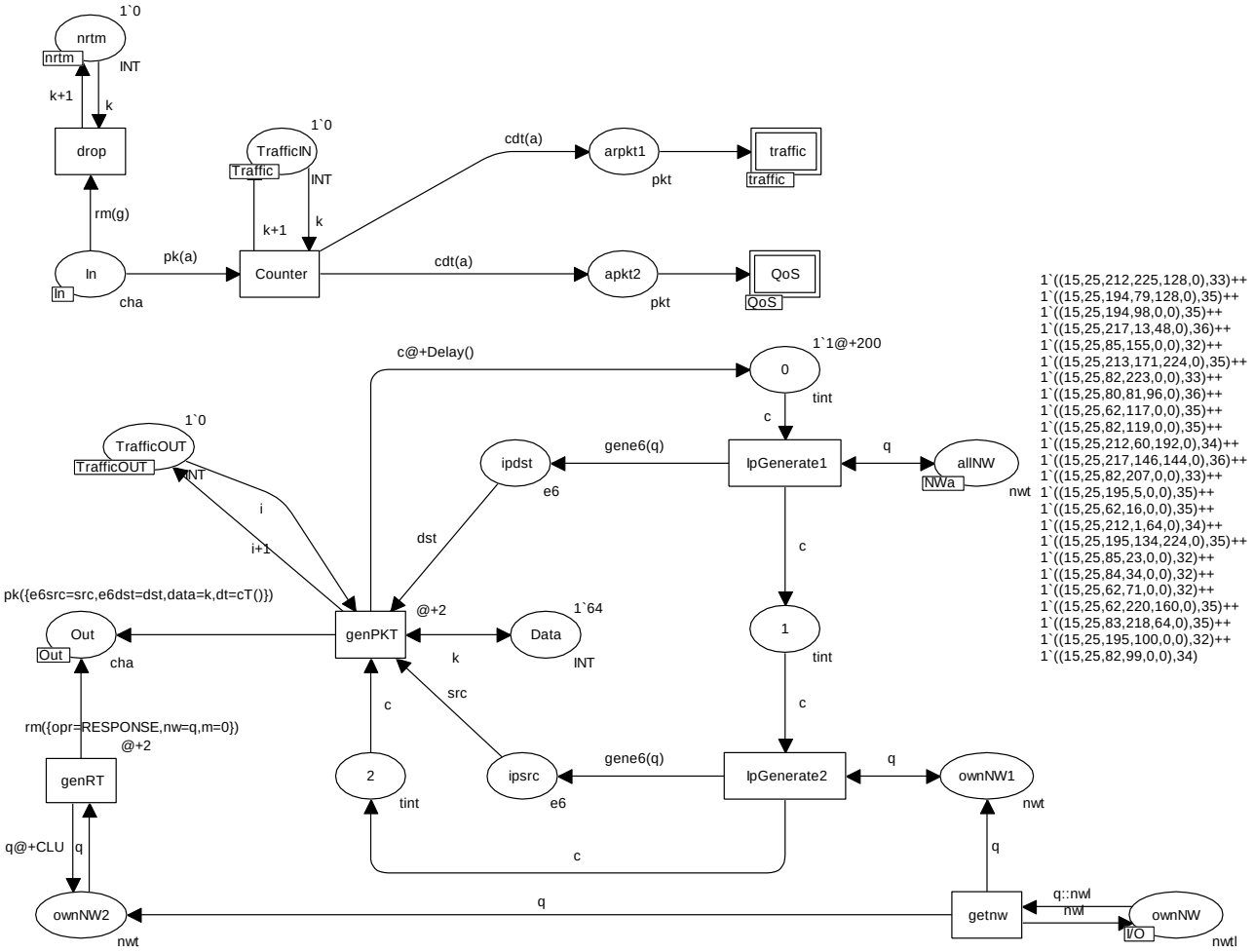


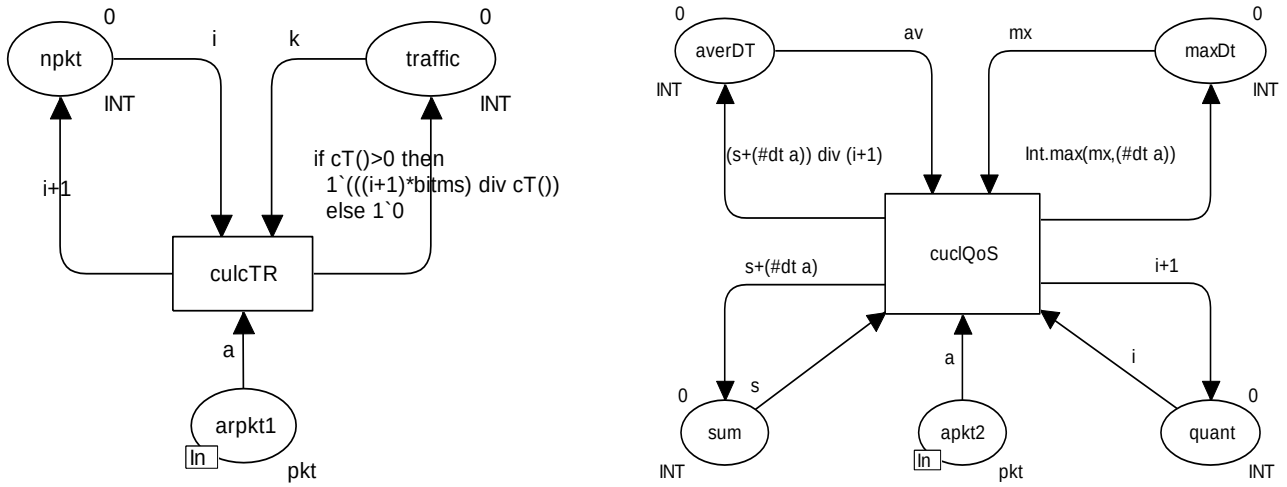
Рисунок 3.7 – Модель термінальної E6 мережі

Генерацію маршрутних поновлень для пропагування власних підмереж виконує перехід genRT. Період генерації CLU відповідає періоду розсилки маршрутних поновлень протоколу E6-RIP. Дані каналу подані за допомогою типу cha, що дозволяє об’єднувати пакети даних pk та повідомлення rm протоколу E6-RIP.

3.3 Вимірювальні фрагменти для потокового трафіку

При моделюванні E6 мереж можуть використовуватися як моделі окремих термінальних пристроїв (серверів, робочих станцій) [13,72], так і моделі термінальних мереж з потоковим поданням трафіку [14], розглянуті в попередньому підрозділі. При моделюванні потокового трафіку використана модель термінальної мережі подана на Рис. 3.7 з вимірювальними фрагментами для оцінки продуктивності і якості обслуговування (Рис. 3.8 а,б). Можлива більш

деталізована оцінка характеристик для окремих пар E6 мереж з використання фрагментів виду [15].



а) Оцінка трафіку

б) Оцінка якості обслуговування (часу доставки пакету)

Рисунок 3.8 – Вимірювальні фрагменти потокового трафіку

Фрагмент Рис. 3.8 а) виконує підрахунок кількості прийнятих пакетів і її перерахування в одиницях біт/с за допомогою константи масштабування bitms; функція cT() повертає поточний модельний час. Фрагмент Рис. 3.8 б) обчислює максимальний і середній час доставки пакетів. Час доставки окремого пакета обчислюється по часовому штампі відправлення і часу одержання, що знаходиться в додатковому полі dt. Позиція sum накопичує суму часів, позиція quant – кількість прийнятих пакетів. Середній час доставки обчислюється в позиції averDT, максимальний час – у позиції maxDT.

Крім того, додаткові лічильники забезпечують оцінку сумарної кількості відправлених, доставлених і загублених пакетів інформації і маршрутних відновлень.

3.4 Дослідження моделі E6 на змінній структурі мережі

Оскільки якість протоколів динамічної маршрутизації пов’язана із здатностями адаптації до змінної структури мережі, особлива увага при моделюванні приділяється засобам тимчасового вимкнення пристроїв.

3.4.1 Побудова моделей E6 мереж із компонентів

Розроблена в підрозділі 3.1 модель комутуючого маршрутизатора КМЕ6 використана в моделюванні низки мереж для оцінки ефективності динамічної маршрутизації E6 мереж. На Рис. 3.10 подано модель фрагмента Європейської магістралі Інтернет [14] за структурною схемою мережі, представленої на Рис. 3.9. Відзначимо, що на схемі зазначені 6 термінальних мереж, що містять E6-адреси в адресному просторі відповідних країн; усього використано 24 E6-мережі, чотири останніх байта E6 адрес відповідають IP-адресам країн, перші два байта обрані нульовими. Магістральна мережа утворена 7-ма комутуючими маршрутизаторами (R1-R7).

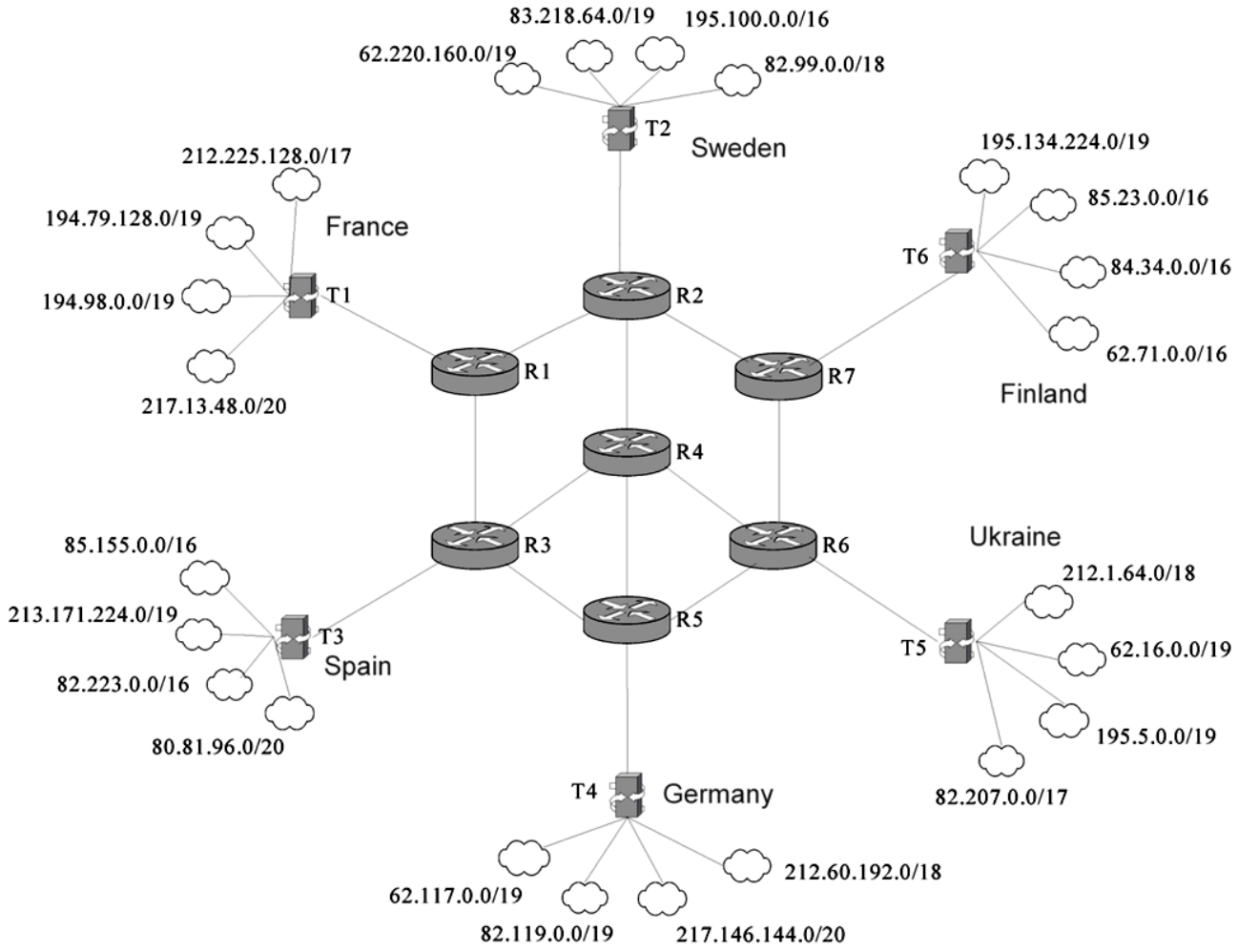


Рисунок 3.9 – Фрагмент Європейської магістралі Інтернет

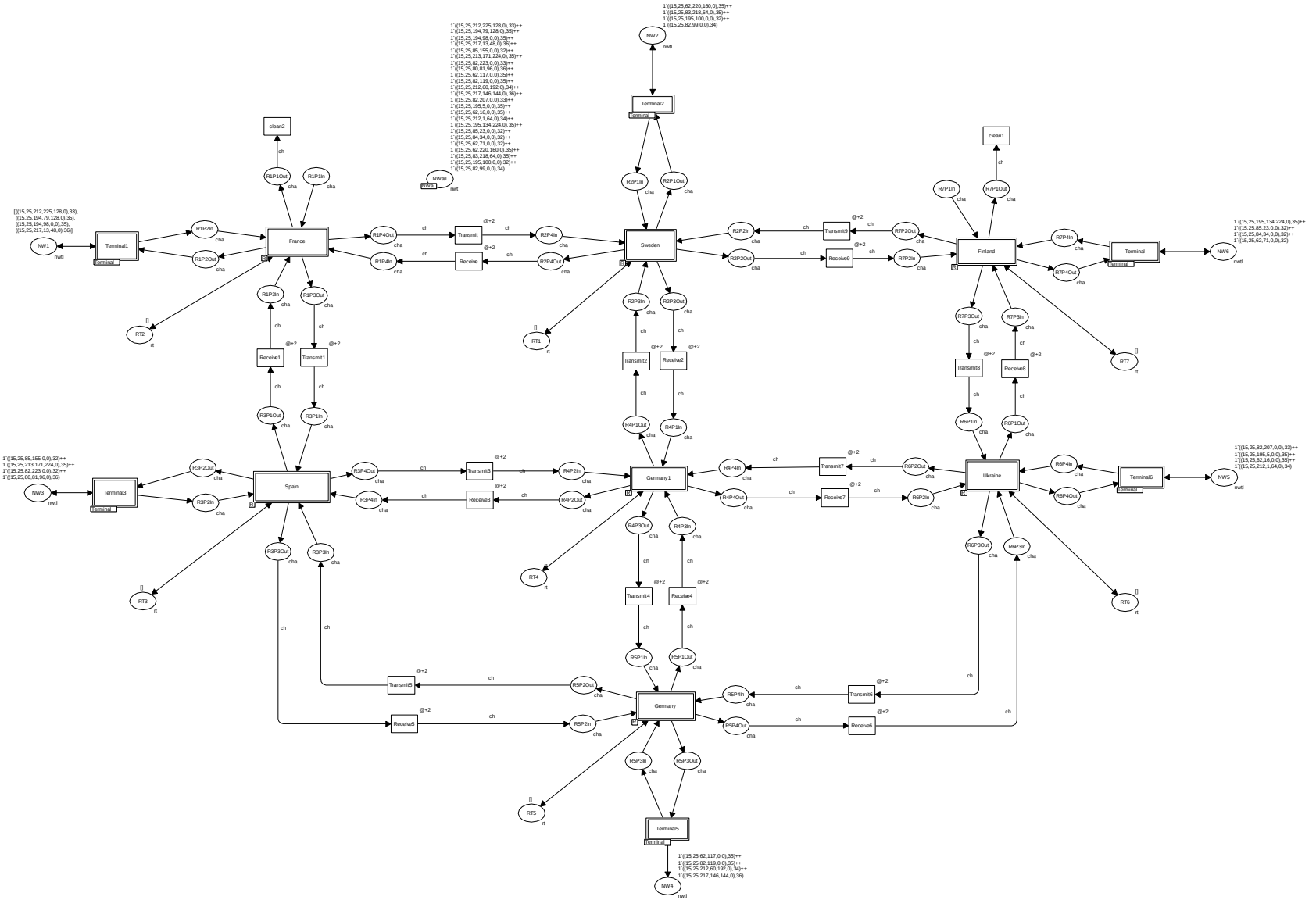


Рисунок 3.10 – Модель Е6 мережі із динамічною маршрутизацією

Для моделювання реалістичного трафіка побудовані спеціальні моделі термінальних мереж Terminal1-Terminal6, описані в підрозділі 3.2. Переходи Transmit*, Receive* моделюють передачу пакетів у каналах без втрати пакетів. Зауважимо, що моделюється повнодуплексний режим передачі пакетів. Тому кожну лінія зв'язку типу „точка-точка" подано парою переходів Transmit*, Receive* для моделювання двох незалежних каналів передачі пакетів для обох напрямків. Позиції NW* містять E6 адреси підмереж відповідних термінальних мереж. Позиція NWall містить адреси всіх E6 підмереж моделі; ця позиція є сполученою і використовується в моделях термінальних мереж для генерації трафіка.

Для спостерігання на головній сторінці моделі таблиці маршрутизації подано як контактні позиції RT*. Але найголовнішою причиною зовнішнього подання таблиць маршрутизації є той факт, що використано той же самий компонент, а саме R для моделювання всіх маршрутизаторів мережі. Дані, якими маршрутизатори відрізняються, повинні бути зовнішніми. Цей факт є також мотивацією для зовнішнього подання адрес власних E6 підмереж термінальних мереж Terminal1-Terminal6, які моделюються одним компонентом Terminal.

Отже технологія побудови головної сторінки моделі міститься в прямому відображенні структурної схеми мережі, за якою розташовуються і з'єднуються компоненти R, Terminal. Зауважимо, що для підвищення швидкості моделювання рекомендується додати переходи для поглинання пакетів на невикористаних портах маршрутизаторів, наприклад переходи clean1, clean2.

3.4.2 Моделі тимчасового вимкнення пристроїв

Засоби тимчасового вимкнення та підрахунку загублених пакетів окремо по пакетах даних та повідомленням E6-RIP значно ускладнюють моделі подані у підрозділі 3.1 на Рис. 3.2, 3.3. Технологію вимкнення зорганізовано на основі номеру порту KME6, який може бути додатнім для робочого порту та від'ємним для виключеного порту.

Розглянемо детально модель порту із засобами вимкнення, зображену на Рис. 3.11. Робота усіх переходів контролюється позицією pnum, яка містить номер порту p. Функціональні переходи прийму та відправки пакетів getrtr, getpkt, putpkt, putrtr мають додаткову умову запуску p>0, яка відключає їх при від'ємному номері порту. Альтернативну умову запуску p<0 мають переходи поглинання пакетів відключеного порту lostrtrin, lostpktin, lostpktout, lostrtrout. При поглинанні пакетів виконується їх підрахування у позиціях lostrtr, lostpkt.

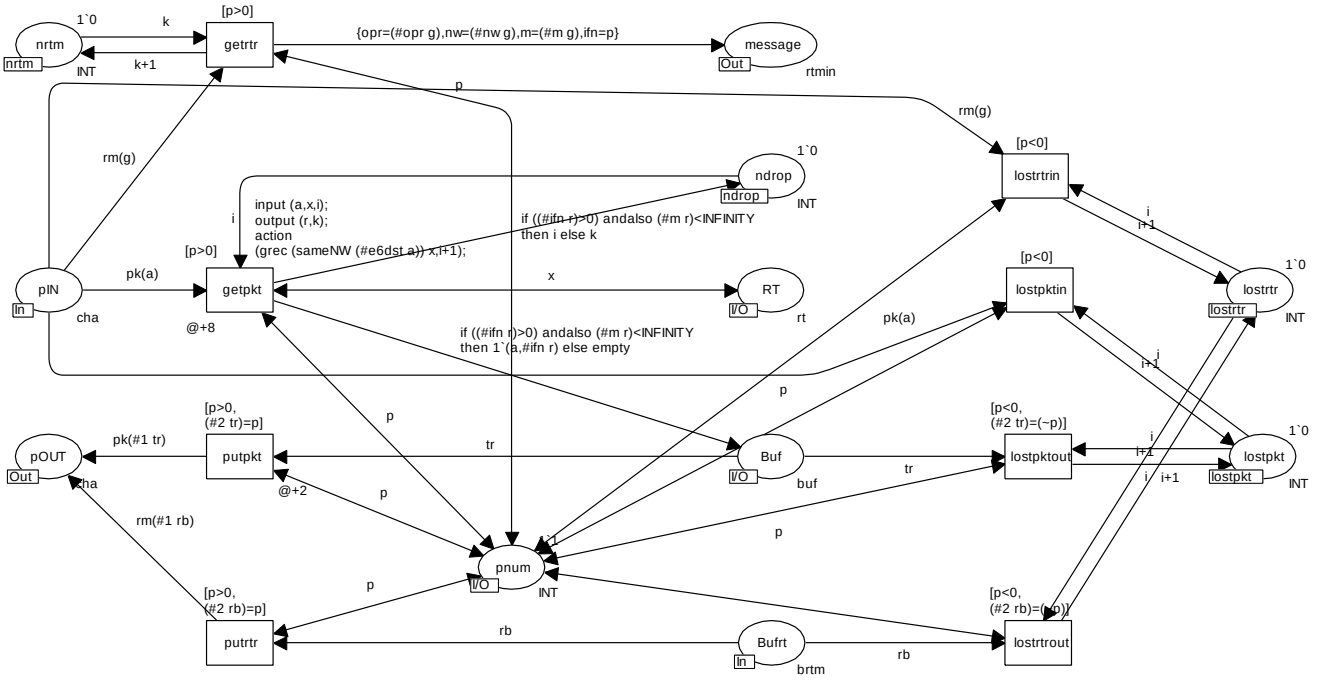


Рисунок 3.11 – Модель порту КМЕ6 із засобами вимкнення

Модель КМЕ6 із вимкненнями зображена на Рис. 3.12 має достатньо складний вигляд зумовленій необхідністю безпосереднього з'єднання кожного з портів із двома переходами Off і On. Перехід Off вимикає всі порти завдяки заміні додатного номеру на від'ємний. Перехід On вмикає всі порти завдяки заміні від'ємного номеру на додатний. Циркуляция фішки у позиціях 1, 2 забезпечує періодичні відключення. Тривалість відключення задає випадкова функція DOffDur(); тривалість нормального функціонування – випадкова функція DOffPer(). Зауважимо, що вмикання та вимикання виконуються одночасно для всіх портів певного КМЕ6, що моделює відключення цілком всього пристрою. Модель порту (Рис. 3.11) дозволяє відключати порти незалежно один від одного але ця можливість не застосована у моделі КМЕ6 (Рис. 3.12). Крім того, при включенні портів за допомогою переходу On в кожний вихідний канал кожного порту p*OUT передається запит inirq на передачу всієї таблиці маршрутизації від сусіда.

3.4.3 Аналіз результатів моделювання Е6 мереж

Розроблені компоненти використані для моделювання конкретних заданих Е6 мереж [14,24,25] разом з моделями термінальних (абонентських) мереж для генерації трафіка (підрозділ 3.2). Модель термінальної мережі Рис 3.3 містить вимірювальні фрагменти для оцінки максимального і середнього часу доставки пакета і трафика (підрозділ 3.3). На мережах з незмінною структурою перевірено правильність заповнення маршрутних таблиць. При запізнілій генерації трафіка загублені пакети відсутні. Для непрямої оцінки якості роботи

протоколу E6-RIP і вибору його параметрів використана кількість загублених пакетів. Для оцінки накладних витрат підраховувалася кількість переданих E6-RIP повідомлень і відношення їхнього розміру до корисного трафіку.

Оскільки адаптація до фіксованої структури є досить тривіальної, основна увага дослідження сконцентрована на поводженні протоколу при змінах структури мережі. Для цих цілей описані моделі доповнені засобами вимикання окремих портів і пристроїв у цілому, а також окремого підрахунку пакетів загублених при відключенні Рис 4.12, 4.13. При включенні пристроїв використані запити на передачу повних таблиць маршрутизації від сусідів. Реалізовано випадкові рівномірно розподілені вимикання пристроїв з періодом OffPer і тривалістю OffDur; можливе застосування інших законів розподілу вимикань.

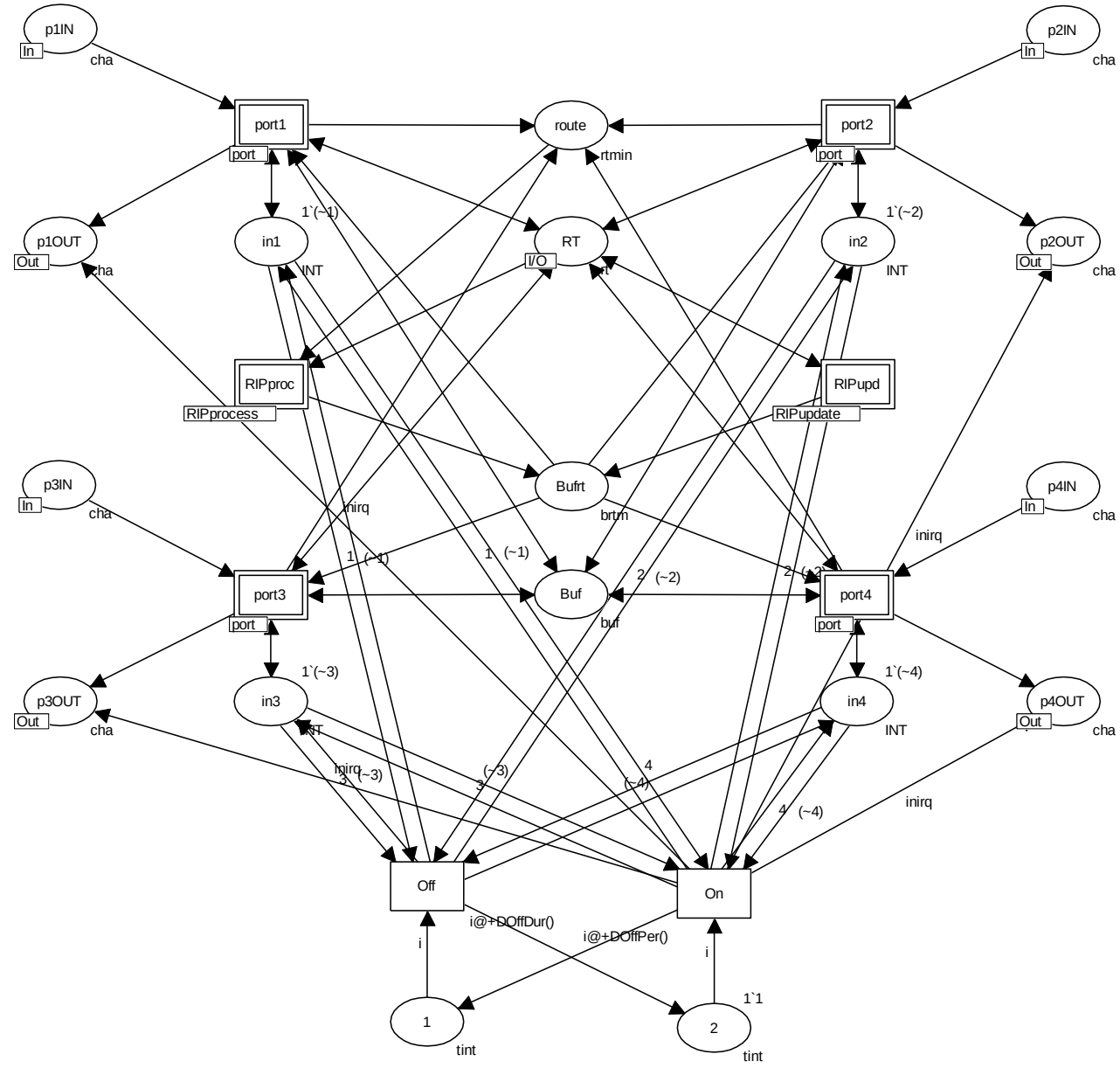
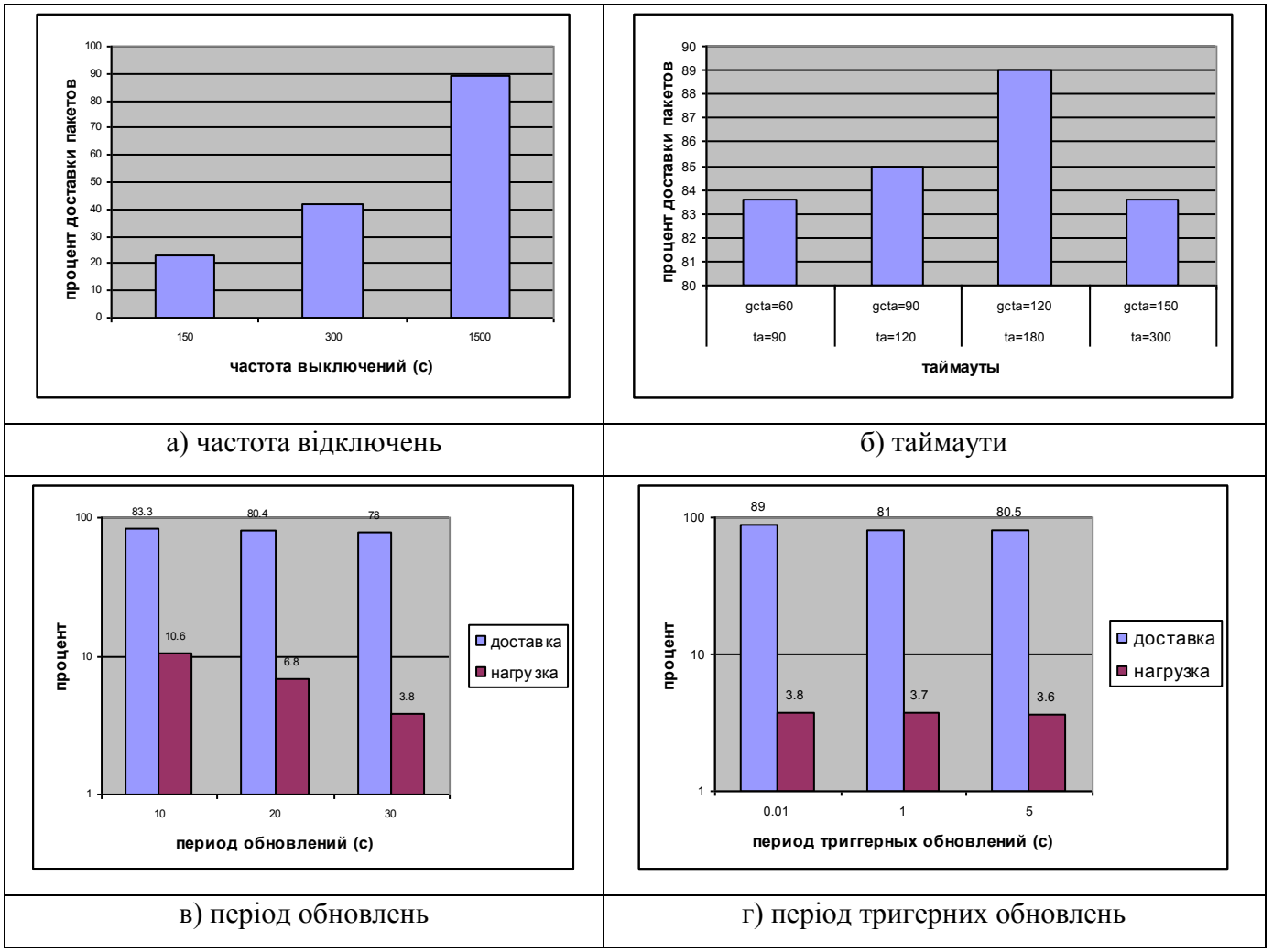


Рисунок 3.12 – Модель КМЕ6 із засобами вимкненням

На Рис. 3.13 представлені результати оцінки впливу параметрів протоколу на відсоток доставлених пакетів і навантаження, створюване динамічною маршрутизацією; при оцінці навантаження на Рис. 3.13 в,г) використана логарифмічна шкала.



а) частота відключень

б) таймаути

в) період обновлень

г) період тригерних обновлень

Рисунок 3.13 – Оцінка впливу параметрів протоколу E6-RIP

Тривалість вимикання OffDur установлена рівної 100-200 с. З Рис. 3.13 а) видно, що період вимикань із тривалістю 200-400 с. (середнє 300 с.) є критичним для роботи мережі; подальші оцінки проводилися для періоду 1000-3000 с. (середнє 1500 с.). При стандартному періоді відновлень, як збільшення, так і зменшення таймаутів приводять до гірших результатів у порівнянні зі стандартними таймаутами (Рис. 3.13 б). Істотний вплив на відсоток доставки пакетів робить період регулярних відновлень (Рис. 3.13 в); однак його зменшення до 10 с. збільшує навантаження протоколу маршрутизації на мережу до 11%. Досить прийнятним є варіант зменшення періоду до 20 с. і пропорційного зменшення таймаутів старіння і збору сміття до 120 і 80 с. відповідно. Вплив періоду тригерних змін (Рис. 3.13 г) є несуттєвим;

зменшення його значень нижче стандартного рівного 1 с. не рекомендовано. Отруєний реверс не дає істотних переваг у порівнянні з простим розщепленням горизонту. Дослідження впливу величини INFINITY утруднено в поточній версії CPN Tools тому що це зв'язано з генерацією й обробкою мережних структур великого розміру, що вимагає значних ресурсів при моделюванні.

Висновки до 3 розділу

1. Розроблено модель комутуючого маршрутизатора E6 із засобами динамічної маршрутизації E6-RIP, яка дозволяє моделювати E6 мережі довільної структури та може бути використана як специфікація КМЕ6 та E6-RIP для подальшої програмно-апаратної реалізації.

2. Розроблено засоби тимчасового вимкнення пристроїв при моделюванні E6 мереж, що дозволяє оцінити здібності протоколу E6-RIP по адаптації до змінної структури мережі.

3. Виконано моделювання низки E6 мереж зі змінною структурою та додатковими вимірювальними фрагментами для оцінки продуктивності, якості обслуговування мережі та службове навантаження створюване протоколом динамічної маршрутизації E6-RIP. Підтверджено прийнятну здібність протоколу E6-RIP по адаптації до змінної структури мережі, а також обрано його основні параметри.

4 МОДЕЛЮВАННЯ PBB МЕРЕЖ

Альтернативним до технології E6 прикладом розв'язання проблеми масштабування Ethernet є технологія магістральних мостів провайдера (PBB), яку запропоновано IEEE у вигляді попереднього стандарту [56]. За допомогою моделювання PBB мереж та порівняльного аналізу результатів із технологією E6 доведено, що технологія E6 має суттєві переваги.

4.1 Огляд технології PBB

Технологія магістральних мостів провайдера [27,43] (Provider Backbone Bridge – PBB) призначена для побудови мереж операторів зв'язку цілком на основі Ethernet. Відповідний стандарт IEEE 802.1ah [56], розробка якого розпочата в 2005 році, підготовлений у чорновому (draft) варіанті, у той час як фірми-виробники вже почали випуск PBB комутаторів і оператори зв'язку приступили до їх експлуатації. У якості Ethernet компонентів мережі проекту 21CN («Мережа 21 століття») Бритиш Телеком вибрав комутатор-маршрутизатор Nortel Metro 8600, а також Metro Ethernet Services Unit серії 1850. Постачання суперпродуктивних PBB комутаторів BlackDiamond 20808 компанії Extreme Networks почалися в Росію.

З появою стандартів 1 Gbps і 10 Gbps Ethernet відкрилися можливості для масового застосування технології Ethernet у магістральних мережах операторів зв'язку, однак технологія 802.3 [55], 802.1D [52] має ряд істотних недоліків, зв'язаних з масштабуємістю, якістю обслуговування і керованістю, перебороти які покликана серія нових стандартів IEEE [43]: 802.1Q – віртуальні мережі [54], 802.1QinQ – багаторівневі віртуальні мережі, 802.1ad – мости провайдера [53], 802.1ah – магістральні мости провайдера [56], 802.1ag – менеджмент мереж, 802.1Qay – інжиніринг трафіка. Зазначені стандарти забезпечують концепцію «Etnernet транспорту» (Carrier Ethernet) [27] для заміни в магістралях операторів як SDH, так і рішень IP-MPLS [3], хоча IETF починає активні спроби інтеграції стандартів MPLS [68,69] і PBB у віртуальному приватному сервісі VPLS [57].

Кадр IEEE 802.1ah [56] інкапсулює кадри IEEE 802.1QinQ і IEEE 802.3. Заголовок кадру IEEE 802.1ah (Рис. 4.1, Табл. 4.1) містить C-MAC – адреси користувача (C-DA, C-SA) і B-MAC – адреси магістралі (B-DA, B-SA). Крім того, передбачена повторна інкапсуляція PBB кадрів для створення багаторівневих магістральних мереж.

Рисунок 4.1 – Формат заголовка кадру IEEE 802.1ah

Таблиця 4.1 – Опис полів заголовка кадру IEEE 802.1ah

Позначення	Назва (оригінальна)	Назва (переклад)
B-DA	Backbone destination address	Магістральна адреса одержувача
B-SA	Backbone source address	Магістральна адреса відправника
B-Tag	Backbone VLAN tag	Магістральний тег віртуальної мережі
I-Tag	Service instance tag	Тег екземпляра сервісу
C-DA	Customer destination address	Користувальницька адреса одержувача
C-SA	Customer source address	Користувальницька адреса відправника
S-Tag	Service provider VLAN tag	Тег віртуальної мережі провайдера
C-Tag	Customer VLAN tag	Тег віртуальної мережі користувача
Data	Data	Дані
FCS	Frame check sequence	Контрольна послідовність кадру

Абстрагуючись від полів віртуальних мереж заголовка кадру, розглянемо взаємодію адресних полів на прикладі мережі, представленої на Рис. 4.2. Нехай комп'ютер X с MAC-адресою AX відправляє кадр комп'ютеру Y з MAC-адресою AY. Формується відповідний кадр 802.3 (802.1ad) з C-DA=AY, C-SA=AX. Кадр доставляється найближчому граничному PBB комутатору PBBX с MAC-адресою ABX. За адресою призначення AY (за допомогою адресних таблиць) комутатор PBBX визначає адреса ABY магістрального PBB комутатора PBBY, до якого підключена мережа, що містить Y. PBBX інкапсулює 802.3 кадр у 802.1ah кадр із указівкою B-DA=ABY, B-CA=ABX і відправляє кадр у магістраль. Магістральні PBB комутатори використовують тільки пари адрес B-DA, B-SA для доставки кадру граничному PBB комутатору PBBY. При одержанні кадру PBBY витягає інкапсульований кадр 802.3 (802.1ad) і виконує доставку кадру комп'ютеру Y, використовуючи пари адрес C-DA, C-SA.

Для заповнення адресних таблиць використовується пасивне прослуховування. Якщо адреса призначення невідома – виконується широкомовлення. На Рис. 4.2 представлена деревоподібна мережа; для роботи на недеревоподібних топологіях стандарт передбачає використання модифікованих алгоритмів покриваючого дерева.

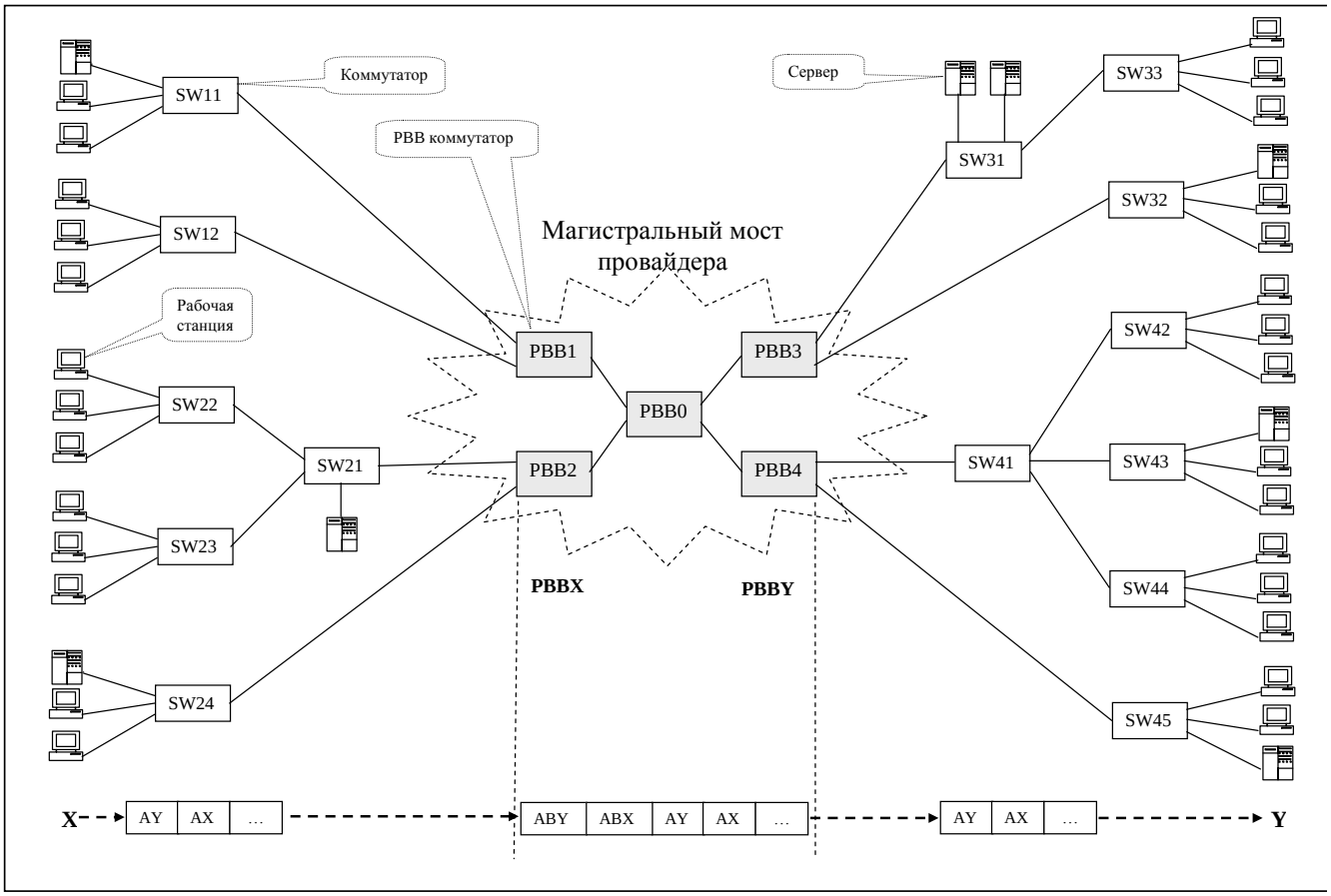


Рисунок 4.2 – Приклад мережі із магістральним мостом провайдера PBB

Перевагою технології PBB є підвищення продуктивності магістралі за рахунок значного скорочення числа записів в адресних таблицях магістральних PBB комутаторів, що містять тільки B-MAC адреси. При цьому ускладнюється робота граничних PBB комутаторів, що виконують відображення C-MAC адрес у B-MAC адреси й інкапсуляцію кадрів. На периферії мережі можуть працювати звичайні 802.1D комутатори.

4.2 Розробка компонентів моделей PBB мереж

Через те що PBB [43] (Додаток А) припускає збільшення довжини заголовків кадрів за рахунок додавання пара MAC-адрес магістральних комутаторів, E6 має певні переваги, обумовлені анулюванням протоколів TCP, UDP, IP, та відповідних заголовків пакетів і протоколів відображення адрес ARP/RARP. Моделі E6 мереж, представлені в Розділі 3, створюють основу для порівняння двох технологій. Однак, повноцінний порівняльний аналіз можливий при побудові досить деталізованих моделей PBB мереж.

Метою дійсної розділу є побудова моделей основних компонентів PBB мереж у формі розфарбованих сітей Петрі [58] в середовищі моделюючої системи CPN Tools [16,35].

4.2.1 Загальна організація моделі PBB мережі

У дійсній роботі при моделюванні PBB технології враховувалася тільки адресна частина заголовків кадрів (Рис. 4.3) без використання тегів віртуальних мереж, крім того, розглядався один рівень ієрархії мостів провайдера і деревоподібна структура мережі. Моделювання тегів віртуальних мереж, багаторівневої ієрархії і роботи алгоритмів покриваючого дерева магістральних мостів є напрямками для перспективних досліджень. Крім того, дослідження повнозв'язних мережних структур (без розподілу на віртуальні приватні мережі), аналогічних до Інтернет, відповідає меті наступного порівняльного аналізу PBB і E6 з погляду переваг використання у всесвітніх мережах.

Для моделювання PBB мереж [53,56] побудовані наступні компоненти:

– модель магістрального PBB комутатора SWBm;

– модель граничного PBB комутатора SWBm-n;

– модель комутатора 802.1D [52] (традиційного) SWBn;

– моделі термінального (абонентського) устаткування: WS – робоча станція, MWS – вимірювальна робоча станція, S – сервер.

Числа m, n позначають кількість B-портів і C-портів відповідно.

За основу при побудові моделі комутатора SWBn (модель порту – port) обрана [73] з динамічним веденням таблиць комутації, модифікована з урахуванням застосування тільки мікросегментованої Ethernet і ведення роздільних черг кадрів по портах. Модель магістрального PBB комутатора SWBm (модель порту – PBBport) має відмінності, зв'язані з обробкою магістральних B-MAC адрес. Найбільш складної є модель граничного PBB комутатора SWBm-n (моделі портів – cport, bport), оскільки вона забезпечує відображення користувальницьких C-MAC і магістральних B-MAC адрес, а також відповідне широкомовлення двох видів.

```
colset mac = INT;
colset mact = mac timed;
colset frm = product mac * mac * nfrm timed;
colset PBBfrm = product mac * mac * mac *
mac * nfrm timed;
colset seg = union f:frm + avail timed;
colset PBBseg = union b:PBBfrm + bavail
timed;
colset swi = product mac * portnum;
colset swita=list swi;
colset PBBswi = product mac * mac * portnum;
colset PBBswita = list PBBswi;
colset qfrm = list frm;
colset pqfrm = product portnum *qfrm;
colset xfrm = union cf:frm + bf:PBBfrm;
colset qxfrm = list xfrm;
colset pqxfrm = product portnum *qxfrm;
```

```
fun eqa a (rr:swi)=((#1 rr)=a);
fun eqaB a (rr:PBBswi)=((#1 rr)=a);
fun eqbaB a (rr:PBBswi)=((#2 rr)=a);
fun grec prd [] = (0,0) | grec prd (q::r) = if prd(q)
then q     else grec prd r;
fun xrec prd [] = [] | xrec prd (q::r) = if prd(q)
then r else q::(xrec prd r);
fun grecB prd [] = (0,0,0) | grecB prd (q::r) = if
prd(q) then q else grecB prd r;
fun xrecB prd [] = [] | xrecB prd (q::r) = if prd(q)
then r else q::(xrecB prd r);
fun Delay() = poisson( Delta );
fun Dexec() = poisson( dex );
fun Nsend() = poisson( nse );
fun cT()=IntInf.toInt(!CPN'Time.model_time)
val TCL=100000000;
```

Рисунок 4.3 – Опис основних типів даних і функцій

При моделюванні трафіка використовувалася концепція взаємодії клієнт-сервер і відповідні компоненти (підрозділ 4.2), у які додані лічильники для оцінки корисного і широкомовного трафіка, а також використані різні закони розподілу випадкових функцій. Слід зазначити, що при моделюванні великомасштабних магістралей надалі доцільно також використовувати моделі потокового трафіка (підрозділ 3.2), щоб абстрагуватися від деталізації периферії мережі.

4.2.2 Моделі PBB обладнання

Моделі мережного устаткування представлені зазначеними раніше моделями комутаторів трьох різних типів: SW4, SWB4, SWB1-2. Загальна організація моделі комутатора має незначні відмінності і може бути розглянута на прикладі комутатора 802.1D [52] (традиційного) SW4. Модель (Рис. 4.4) набирається клонуванням необхідної кількості моделей портів port.

Кожен порт ідентифікується унікальним номером (myport*). Для взаємодії портів використовуються спільні дані, збережені в пам'яті комутатора. У якості найбільш простої розглядається архітектура з обов'язковим буферуванням кадрів. Кадр, що надійшов у вхідний канал порту A, розміщається для тимчасового збереження в буфері Buf; при цьому за допомогою таблиці комутації Sw визначається номер порту B для перенаправлення кадру. Якщо адреса призначення кадру не зазначена у таблиці, порт виконує широкомовлення – кадр перенаправляється в усі порти комутатора, крім порту A. Вихідний канал порту B витягає кадр із буфера і передає його у відповідний сегмент. Загальна кількість портів комутатора nport і номер власного порту myport* використовуються в широкомовленні. Позиція timer містить MAC адреси, зазначені в таблиці Sw, разом з часовими мітками (тип даних mact); перехід ClrSwTa забезпечує видалення відповідного запису таблиці Sw після закінчення інтервалу часу старіння запису (константа TCL); рекурсивна функція xrec виконує видалення запису з таблиці, представленої змінною x; функція eqa виконує порівняння адрес. Порт реалізує пасивне прослуховування трафіка (адрес відправника) з метою заповнення таблиці новими записами.

Слід зазначити, що в буфері Buf типу qpfrm організовані роздільні FIFO черги кадрів по портах комутатора у відповідності до стандартів. Початкове маркірування створює 4 порожні черги (списки); заголовок черги дорівнює номеру відповідного порту.

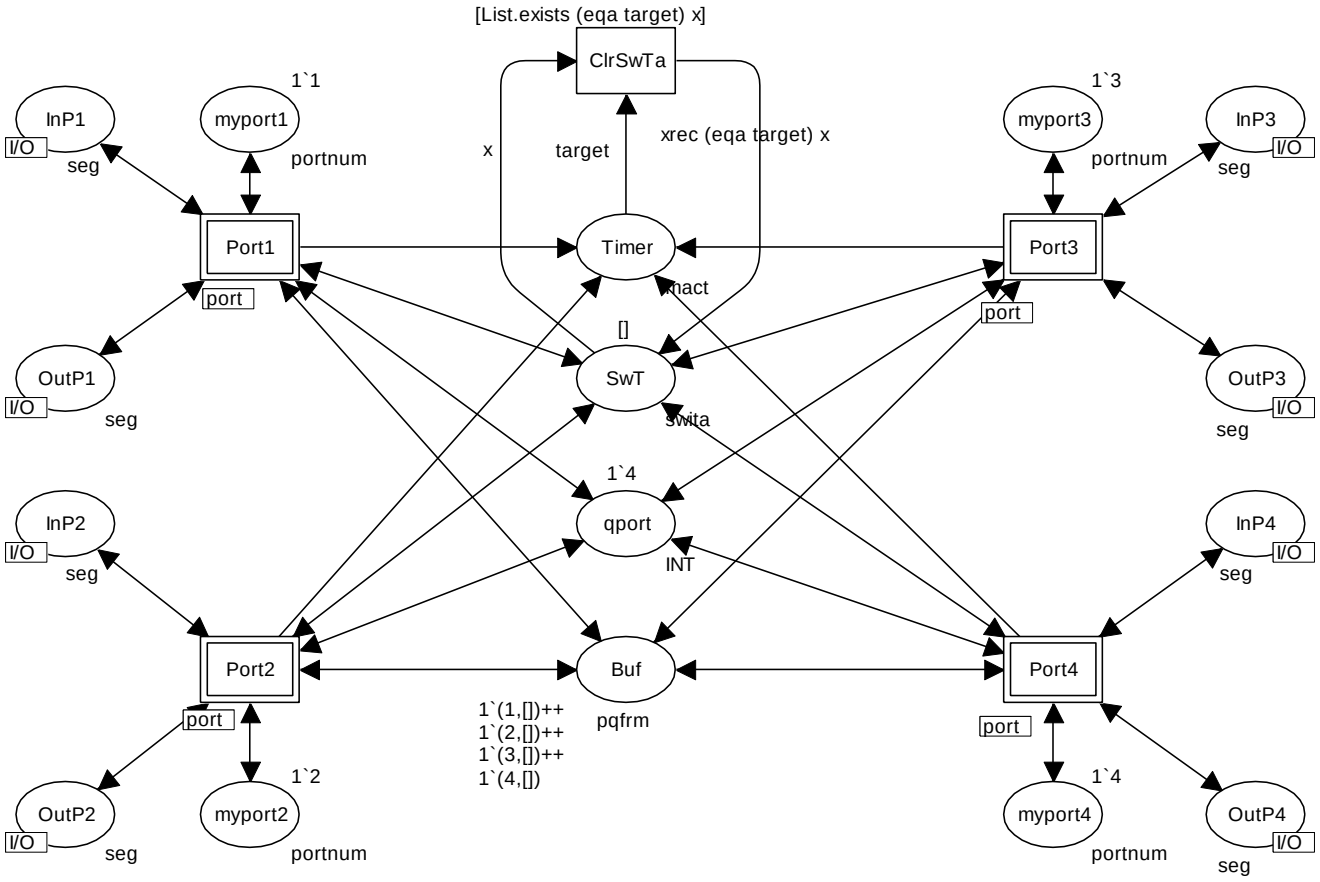


Рисунок 4.4 – Модель комутатора 802.1D (SW4)

1) Модель порту комутатора (port)

Модель порту port комутатора 802.1D представлена на Рис. 4.5. Тип seg використаний для опису каналу сегмента Ethernet, що може бути або вільним (константа avail), або зайнятим передачею кадру f. Тип frm використаний для опису кадру f, що складається з адреси відправника src, адреси одержувача dst і номера кадру nf (опис абстрагується від умісту кадру). Тип swita описує таблицю комутації як список записів swi, що складаються з адреси призначення dst і номера порту pnum. Рекурсивна функція grec вибирає запис таблиці комутації. Тип pqfrm буфера Buf описує занумеровані по портах черги qfrm кадрів frm. Надписи дуг додавання і вилучення, що будуть описані далі, реалізують FIFO дисципліну черг.

Кадр надходить у вхідний канал порту PortIn, при цьому адреса відправника src може бути або новою (перехід NewSrc), або відомою (перехід OldSrc). Перехід NewSrc поповнює таблицю комутації новим записом, у якому зазначена адреса відправника src і номер порту m (поточний порт комутатора). Кадр розміщається в проміжній позиції Aux1, потім аналізується адреса одержувача кадру dst, що може бути або новою (перехід NewDst), або відомою (перехід OldDst). Перехід OldDst розміщає кадр у буфері Buf із указівкою знайденого номера вихідного

порту. Перехід NewDst розміщає кадр у проміжній позиції Aux2 і запускає широкомовлення. Позиція pnum використовується для послідовної (i:=i+1) нумерації портів широкомовлення. Перехід Broad виконує широкомовлення доти, поки не вичерпаються номери всіх портів (i<=q); потім при (i>q) запускається перехід clean, що очищає проміжні позиції і повертає ознаку приступності avail у сегмент. Перевірка (i<>m) у надпису дуги широкомовлення Broad->Buf виключає широкомовлення у власний порт.

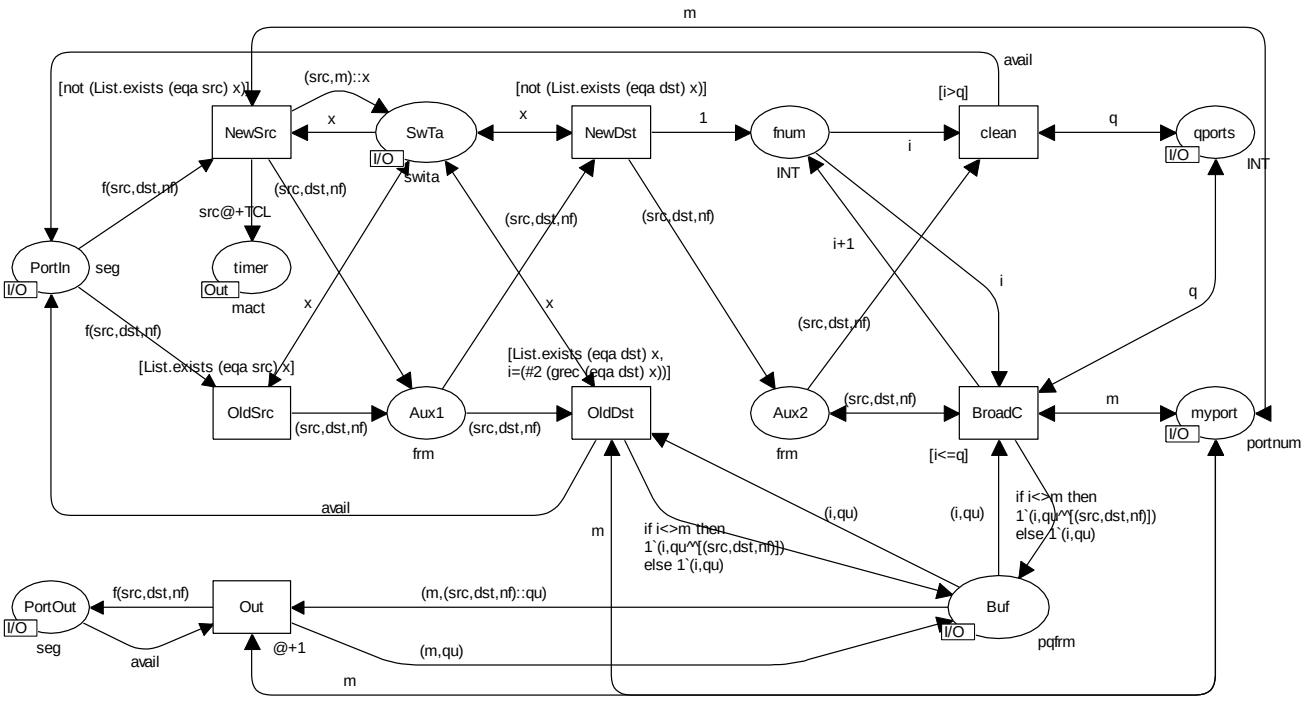


Рисунок 4.5 – Модель порту комутатора 802.1D (port)

Вихідний канал порту вибирає кадри з буфера Buf, перенаправлені в поточний порт (m), і передає їх у сегмент за допомогою переходу Out, що очікує і видаляє ознаку приступності сегмента avail.

Розглянемо більш докладно роботу з чергами кадрів у буфері Buf. Додавання запису виконується в хвіст черги порту i; для цього з позиції Buf витягається відповідна черга за допомогою надпису дуги (i,qu), потім кадр f додається в хвіст черги за допомогою надпису дуги (i,qu^^[f]). Витяг запису виконується з голови черги порту m; для цього з позиції Buf витягається черга з виділеним першим кадром за допомогою надпису дуги (m,f::qu), потім черга без першого елемента повертається за допомогою надпису дуги (m,qu). Операція ^^ виконує конкатенацію черг; операція :: виділяє головний елемент.

2) Модель порту магістрального PBB комутатора (PBBport)

Модель порту PBBport магістрального PBB комутатора представлена на Рис. 4.6. Робота порту магістрального PBB комутатора багато в чому аналогічна роботі порту звичайного комутатора представленого на Рис. 4.5 з тією різницею, що замість C-MAC адрес використовуються B-MAC адреси 802.1ah кадру [56].

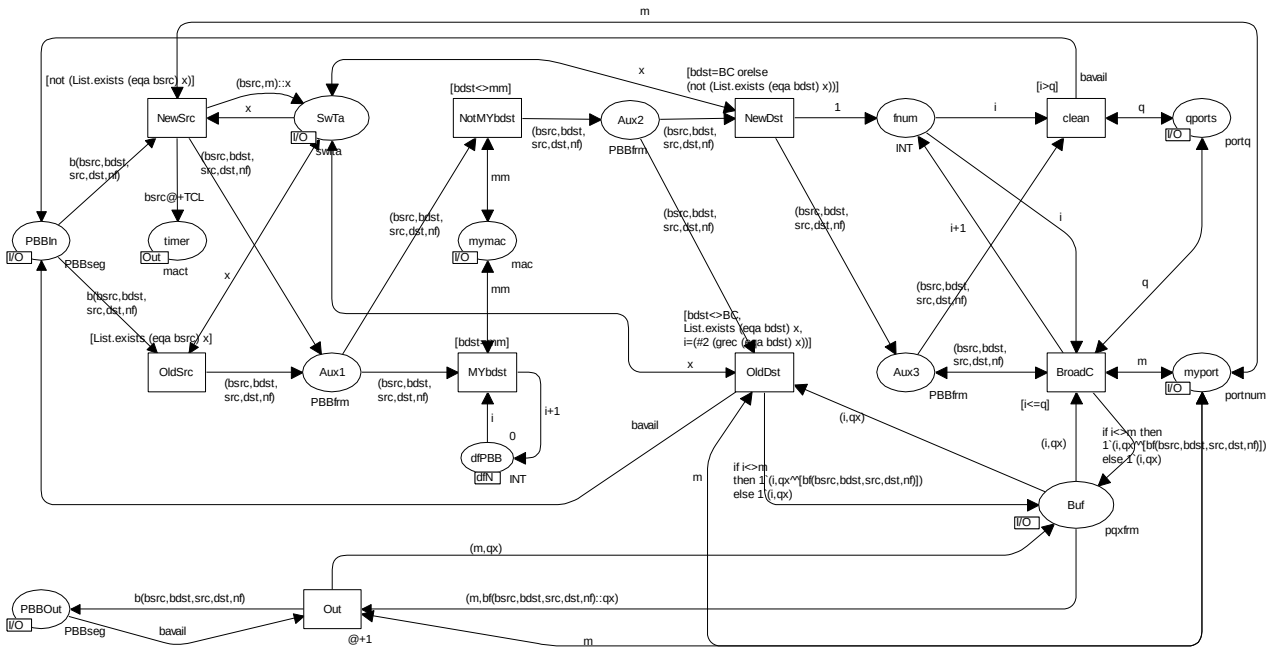


Рисунок 4.6 – Модель порту магістрального PBB комутатора (PBBport)

Тип PBBseg використаний для опису каналу PBB сегмента, що може бути або вільним (константа bavail), або зайнятим передачею кадру b. Тип PBBfrm використаний для опису кадру b, що складається з адреси відправника магістралі bsrc, адреси одержувача магістралі bdst і інкапсульованого 803.3 кадру frm. Таблиця комутації містить тільки адреси магістралі B-MAC. Тип pqxfrm буфера Buf описує черги кадрів xfrm портів; тип xfrm являє собою об'єднання кадрів cf типу frm або кадрів bf типу bfrm. Магістральний PBB комутатор обробляє тільки bfrm кадри; можливість об'єднання кадрів використана в моделях граничних комутаторів.

Основні відмінності в роботі порту пов'язані з обробкою кадру типу PBBfrm замість frm і використанням адрес bsrc, bdst замість адрес src, dst відповідно. Крім того, перехід NotMYbdst установлює, що кадр не адресований поточному комутатору, а перехід MYbdst моделює обробку (службового) кадру, адресованого поточному комутатору, шляхом поглинання кадру та інкременту лічильника dfn у позиції dfPBB.

3) Моделі портів граничного PBB комутатора (cport, bport)

Основна відмінність портів граничного PBB комутатора складається в обробці як C-MAC адрес, так і B-MAC адрес і веденні таблиць комутації, що забезпечують відображення C-MAC адрес у B-MAC адреси. Крім того, виконується широкомовлення двох видів: по C-портах і по B-портах, що передають кадри різних типів.

Для опису елементів черг внутрішнього буфера використаний тип даних об'єднання xfrm, що може зберігати або кадр cf типу frm, або кадр bf типу PBBfrm. Для плоского реляційного представлення багаторівневих таблиць комутації і відображення адрес використаний тип даних PBBswi, що містить адресу призначення dst, адресу призначення магістралі bdst і номер порту pnum. Дублювання полів bdst у декількох записах має перевагу швидкого пошуку повної інформації за ключем src.

3.1) C-порт

Модель C-порту cport граничного PBB комутатора представлена на Рис. 4.7. Основна відмінність від портів port, PBBport (Рис. 4.5, 4.6) складається в поповненні таблиці комутації записами, що містять крім src, магістральну адресу bsrc, що збігається з власною B-MAC адресою. Крім того, модифікована функція рекурсивного пошуку grec і функція порівняння адрес eqa для обробки записів типу PBBswi; використані як друге (#2) так і третє (#3) поля таблиці PBBsw; виконується додаткова перевірка відомої адреси призначення за допомогою переходів OldDstMy, OldDstNotMy на приналежність власної мережі. Якщо відома адреса призначення dst знаходиться у власній мережі (перехід OldDstMy) формується запис cf, що потім перенаправляється в C-порт. Якщо відома адреса призначення dst не знаходиться у власній мережі (перехід OldDstNotMy) формується запис bf, що потім перенаправляється в B-порт, при цьому з таблиці визначається не тільки номер порту призначення (#3 (grec (eqa dst) y)), але також магістральна адреса призначення (#2 (grec (eqa dst) y)). Широкомовлення (перехід Broad) розрізняє порти за допомогою позиції nPBBp, що зберігає номер першого B-порту; порти нумеруються послідовно: спочатку усі C-порти, потім усі B-порти. Тому зазначена в надпису дуги Broad->Buf умова i<pbp виділяє тільки C-порти, його альтернатива (then) – тільки B-порти; у залежності від цього формується cf або bf запис у буфері відповідно. Для широкомовлення в магістралі як адреса призначення bdst використовується константа BC, рівна 255.

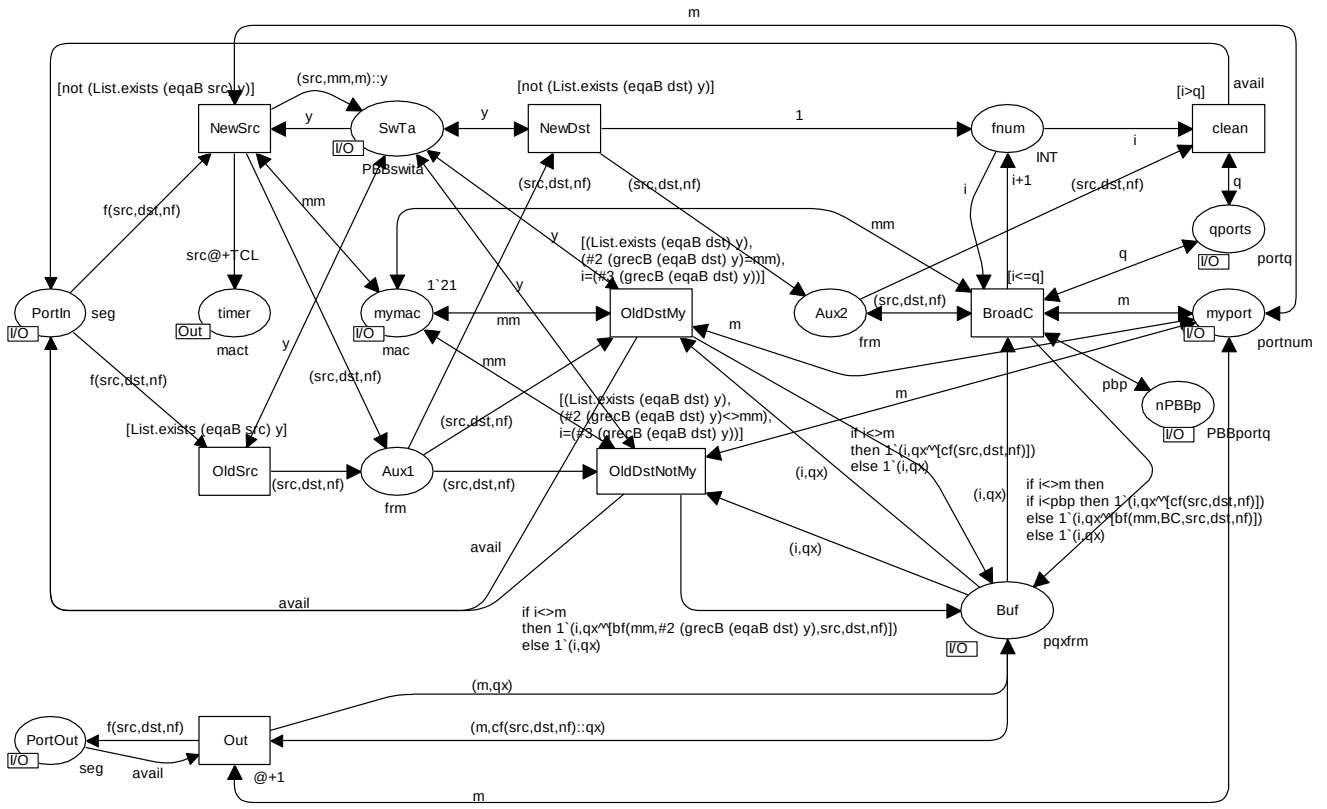


Рисунок 4.7 – Модель С-порту граничного PBB комутатора (cport)

3.2) B-порт

Модель B-порту граничного PBB комутатора bport представлена на Рис. 4.8. Вона є найбільш складної, тому приведемо її докладний опис. Пасивне прослуховування (переходи NewSrc, OldSrc) поповнює таблицю SwTa записами (перехід NewSrc), що містять як користувальницьку src, так і магістральну bsrc адреси з поточного кадру. Потім аналізується магістральна адреса призначення bdst; альтернативи представлені переходами: bdstMy – власний bdst поточного комутатора, bdstBC – широкомовний bdst, bdstNotMy – адреса bdst деякого іншого PBB комутатора. Потім виконується пошук запису в таблиці: ключ – адреса призначення dst для власного bdst (переходи dstNew, dstOld), ключ – адреса bdst для чужого bdst (переходи bdstNew, bdstOld). В обох випадках при вдалому завершенні пошуку кадр перенаправляється в буфер (переходи bdstOld, dstOld); у першому випадку (перехід bdstOld) – без зміни, у другому випадку (перехід dstOld) – витягається інкапсульований 802.3 кадр. Інші варіанти перевірок (переходи dstNew, bdstBC, bsdtNew) приводять до запуску широкомовлення (перехід Broad). У такий спосіб формується множина з п'яти альтернатив: bsdtMy&dstNew, bdstMy&dstOld, bdstBC, bdstNotMy&bdstNew, bdstNotMy&bdstOld. Крім того, спеціально обробляється (перехід wrong) випадок можливої помилки: кадр адресований поточному

комутатору (bdstMy), адреса dst є присутньою у таблиці, але запис таблиці містить bdst відмінний від адреси поточного комутатора.

Для правильного формування широкомовлення використані додаткові булеві ознаки в позиціях: BCcport – широкомовлення по С-портах, BCbport – широкомовлення по В-портах. Кожна з трьох альтернатив широкомовлення формує власний набір ознак: bsdtMy&dstNew – BCcport=true&BCbport=false, bdstBC – BCcport=true&BCbport=true, bdstNotMy&bdstNew – BCcport=false&BCbport=true. У залежності від комбінації ознак надпис дуги Broad->Buf формує кадри широкомовлення перенаправлені в С-порти і В-порти.

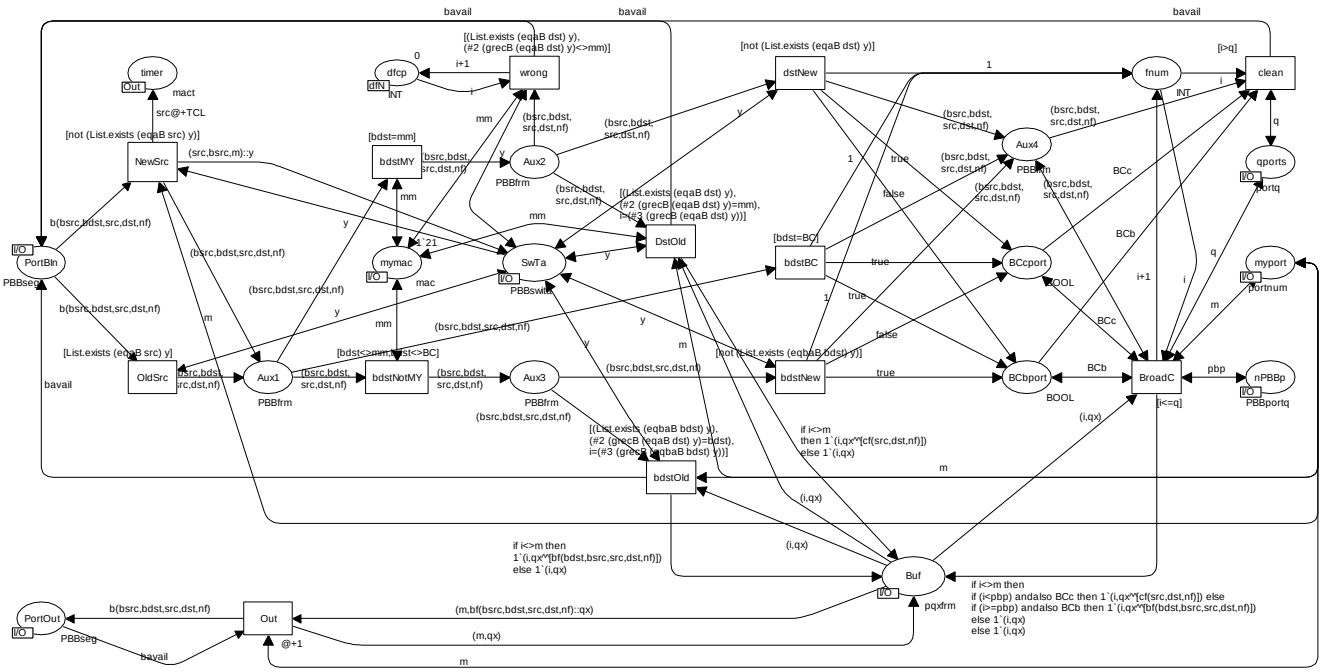


Рисунок 4.8 – Модель В-порту граничного PBB комутатора (bport)

Слід зазначити, що тому що елементи черг буферу представлені об'єднанням, вихідні канали С-портів (перехід Out) витягають з нього запис cf, В-портів – запис bf, і передають у сегменти (типу seg або PBBseg) відповідні кадри.

4.3 Моделі термінального обладнання взаємодії клієнт-сервер

У дійсному розділі використані моделі термінального устаткування, представлені в [13,72] та доповнені засобами витрачання кадрів адресованих до інших пристроїв, а також засобами підрахування кількості відправлених, отриманих та витрачених кадрів.

Робоча станція WS (Рис. 4.9) періодично генерує запити серверам S (Рис. 4.10); розподіл часу між запитами заданий випадковою функцією Delay(). Сервер виконує запит робочої станції і повертає випадкове число кадрів відповіді; розподілення числа кадрів задано випадковою функцією Nsend(); розподілення часу обробки запиту задано випадковою функцією Dexec(). Аналізувалися результати застосування різних законів розподілу випадкових величин: рівномірне, Пуассона, Ерланга. При необхідності можуть бути вивчені більш складні схеми взаємодії в системах клієнт-сервер, обумовлені специфікою розв'язуваних задач.

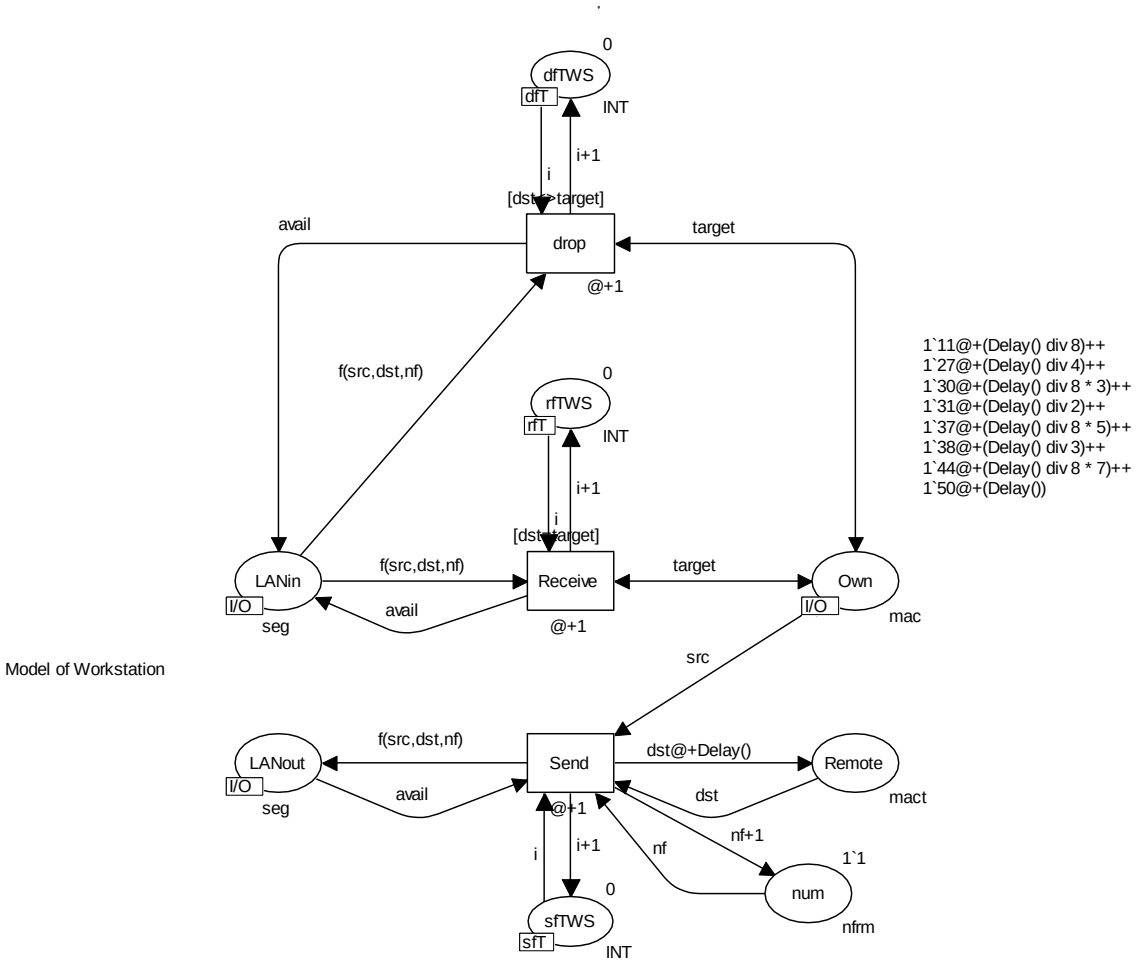


Рисунок 4.9 – Модель робочої станції

У моделі термінального устаткування додані лічильники, представлені сполученими позиціями: SndFrm виду sf – лічильник відправлених кадрів, RcvFrm виду rf – лічильник прийнятих кадрів, DrpFrm виду df – лічильник загублених кадрів. Для зручності оцінки характеристик моделі лічильники винесені на головну сторінку (Network). Крім того, вимірювальні робочі станції MWS виконують оцінку часу відгуку мережі безпосередньо в процесі моделювання [13].

У моделі WS робочої станції (Рис. 4.9) позиції LANin і LANout моделюють вхідні і вихідні канали сегмента, що працює в полнодуплексном режимі. Робоча станція слухає сегмент за допомогою переходу Receive, що одержує кадри з адресами призначення, рівними власній адресі робочої станції (dst=target), збереженому в позиції Own. Обробка кадрів представлена їхнім поглинанням. Робоча станція посилає періодичні запити до сервера за допомогою переходу Send. Адреси серверів зберігаються в позиції Remote. Після посилання запиту використання адреси сервера блокується на випадковий інтервал часу, заданий функцією Delay(). Посилання кадру виконується лише в тому випадку, якщо сегмент мережі вільний, що реалізовано перевіркою позиції LANout на наявність фішки avail. Робоча станція може взаємодіяти з декількома серверами, зберігаючи їхньої адреси в позиції Remote.

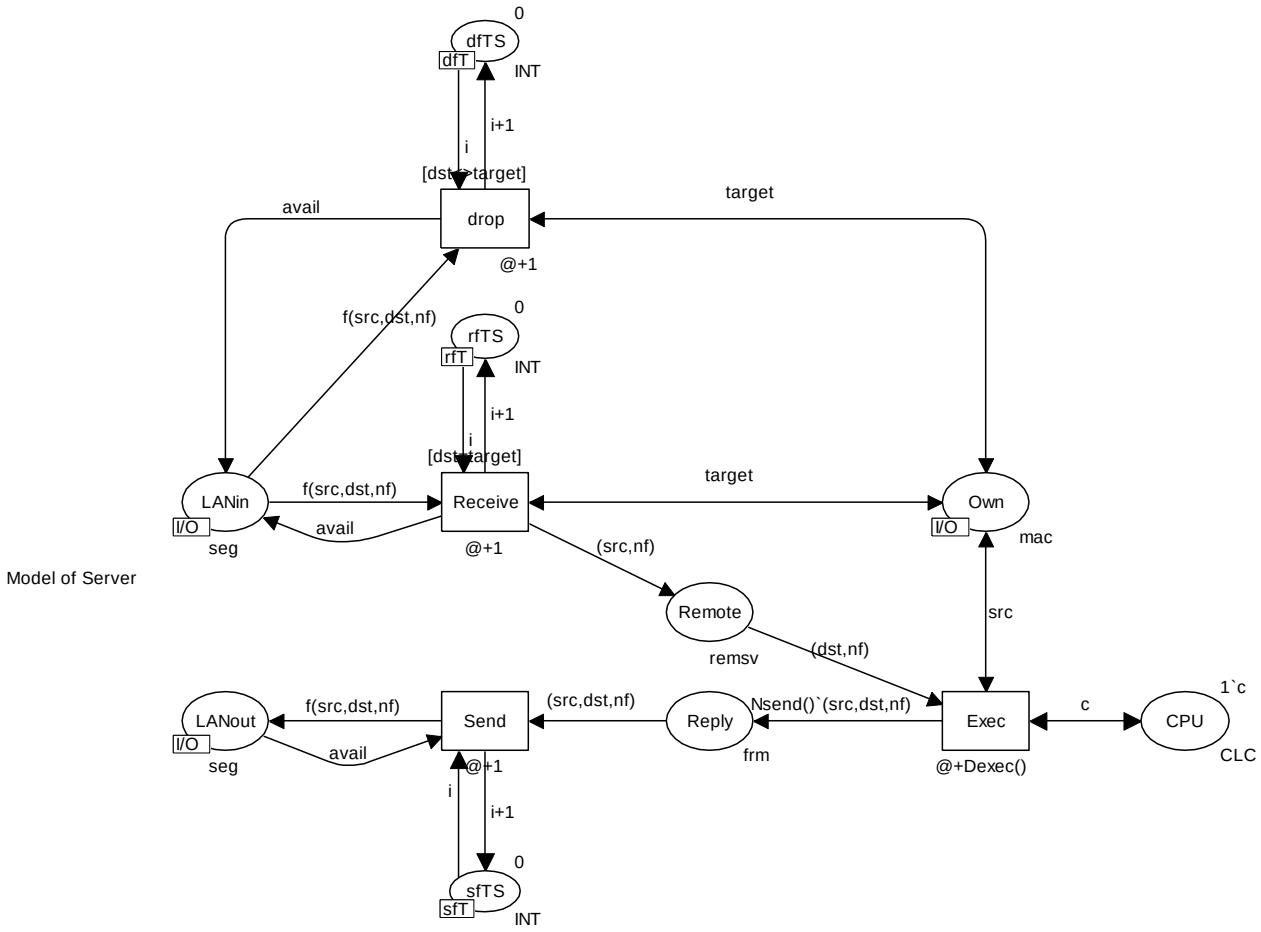


Рисунок 4.10 – Модель сервера

Помітимо, що третє поле кадру з ім’ям nfrm не використовується звичайною робочою станцією. Робоча станція лише задає йому значення рівне одиниці. Це поле використане в моделі спеціальної вимірювальної робочої станції MWS. Копії описаної моделі WS представляють робочі станції WS1-WS24. Для унікальної ідентифікації кожної робочої станції

використана контактна позиція Own. Ця позиція представлена в загальній моделі мережі (Рис. 4.12) позиціями a* й містить MAC-адресу комп'ютеру.

У моделі сервера S (Рис. 4.10) прослуховування сегмента мережі аналогічно моделі робочої станції, відмінність полягає в тому, що адреса відправника кадру src і порядковий номер запиту nf зберігаються в позиції Remote. Перехід Exec моделює виконання запиту робочої станції сервером. У результаті виконання запиту сервер генерує випадкове число Nsend() кадрів відповіді, що зберігаються в позиції Reply. Потім ці кадри по одному передаються в мережу переходом Send. Помітимо, що номер запиту nf, збережений у позиції Remote, використовується для ідентифікації відповіді тим же самим номером, що й запит.

4.4 Модель вимірювальної робочої станції

Модель вимірювальної робочої станції (MWS) зображена на Рис. 4.11. Власне кажучи, вона являє собою розглянуту в попередньому підрозділі модель робочої станції WS, доповнену вимірювальними елементами, що утворять фрагмент мережі, виділений малиновим кольором.

Нагадаємо, що розширені сіті Петрі є універсальною алгоритмічною системою [20,26] і дозволяють описувати не тільки досліджуваний об'єкт, але також додаткові алгоритми, призначені як для керування об'єктом [29], так і для організації вимірів характеристик функціонування [13]. При цьому алгоритм може бути сформований із фрагментів мереж [26,29], що представляють логічні, арифметичні операції, альтернативні і паралельні процеси, характерні для сучасних мов програмування. Детальні описи таких алгоритмів, цілком зібрані з фрагментів сітей можуть бути досить громіздкими. CPN Tools [16] дозволяє одержати розумний компроміс між єдністю інструментальних засобів і компактністю представлення моделі. Убудована мова ML дає можливість доповнити сіть Петрі операторами мови програмування, асоційованими як з дугами, так і з переходами сіті. При цьому необхідні величини можуть зберігатися як у позиціях сіті Петрі, так і в змінних мови ML.

Розглянемо більш детально вимірювальні елементи, представлені на Рис. 4.11. Кожен запит робочої станції нумерується унікальним числом, що міститься в позиції num. Час відправлення запиту зберігається в позиції nSnd. Функція cT() визначає поточний модельний час. Позиція nSnd запам'ятовує пари: номер запиту (кадру) nf і час відправлення запиту в мережу.

Позиція return запам'ятовує часові штампи усіх кадрів, що повернулися. Як час відгуку мережі розглянутий інтервал часу між посилкою запиту й одержанням першого з кадрів відповіді. Це значення зберігається в позиції NRTs для кожного обробленого запиту. Перехід

isFirst розпізнає перший кадр відповіді. Надпис дуги, що з'єднує перехід isFirst із позицією NRTs, обчислює час відгуку (t2-t1).

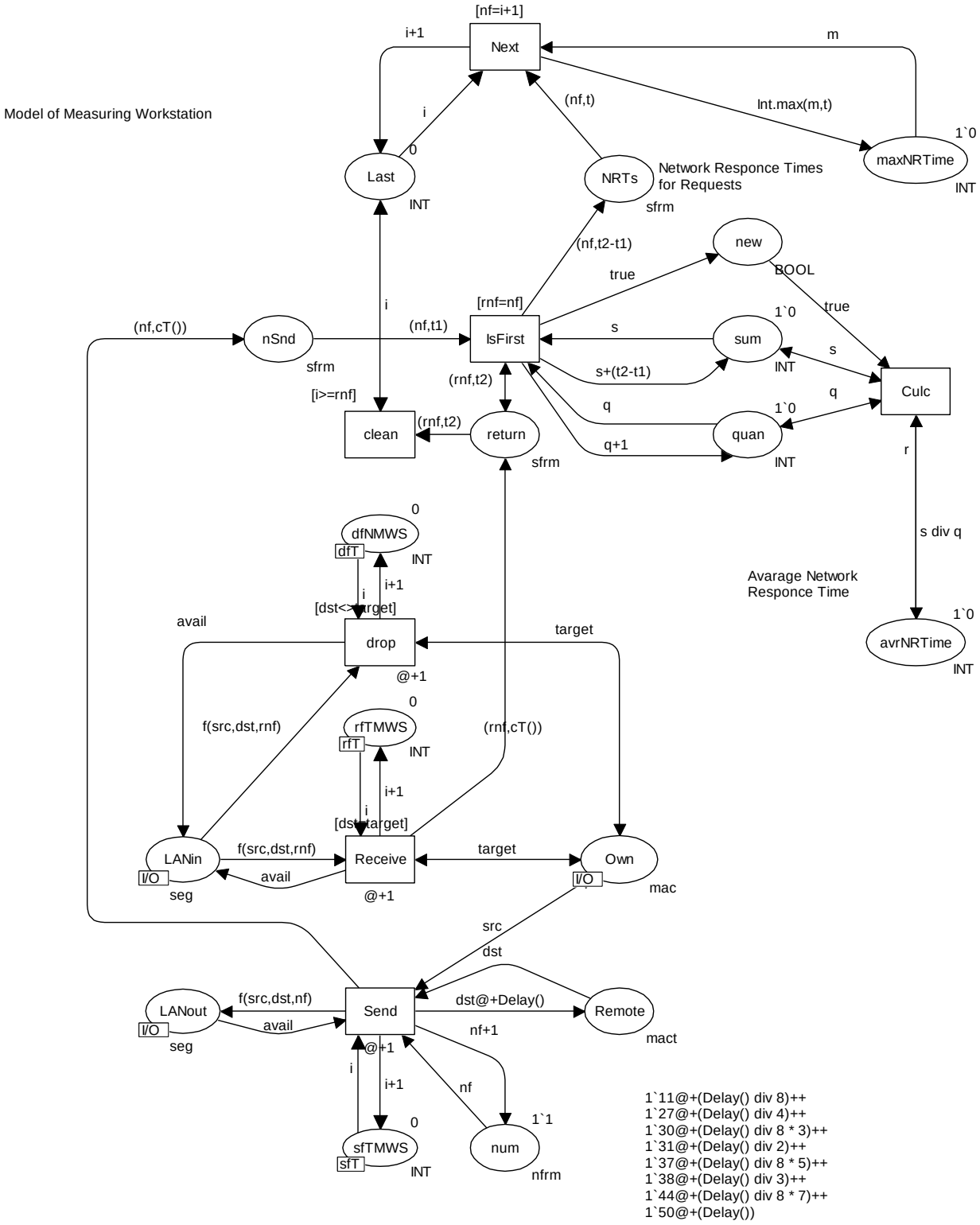


Рисунок 4.11 – Модель вимірювальної робочої станції

Частина вимірювального фрагмента, що залишилася, обчислює середній час відгуку. Позиції sum і quant накопичують суму часів відгуку і кількість прийнятих відповідей відповідно. Прибуття нової відповіді розпізнається позицією new і ініціює переобчислення середнього часу відгуку за допомогою переходу Culc. Результат зберігається в позиції NRTime.

4.5 Дослідження моделей PBB мереж

Метою дійсного підрозділу є оцінка ефективності технології PBB за допомогою моделювання роботи магістральних PBB мереж. Розроблені компоненти використано для побудови і дослідження моделі мережі, зображеної на Рис. 4.12. Вимірювальні робочі станції MWS забезпечують оцінку часу відгуку мережі; лічильники, представлені сполученими позиціями, винесеними на головну сторінку моделі, забезпечують оцінку корисного і широкомовного трафіка.

4.5.1 Побудова моделі PBB мережі із компонентів

Модель мережі, зображеної на Рис. 4.2, представлена головною сторінкою моделі Network на Рис. 4.12. і моделями використаних компонентів на Рис. 4.4-4.11. Головна сторінка побудована на основі принципу прямого відображення структурної схеми мережі. Для моделювання PBB магістралі використаний один 4-х портовий магістральний PBB-комутатор PBB0 типу SWB4 і 4 граничних 3-х портових PBB-комутатори PBB1-PBB4 типи SWB1-2 з одним B-портом і двома C-портами. Для моделювання периферійних мереж використані 14 4-х портових комутаторів типу SW4. Модель термінального (абонентського) устаткування представлена 24 робочими станціями WS, 4 вимірювальними робочими станціями MWS і 8 серверами S.

Описи основних типів даних і функцій моделі приведені в підрозділі 4.2. На головній сторінці моделі використані наступні типи позицій: mac – MAC-адреса, seg – звичайний сегмент, PBBseg – PBB сегмент; як ознаку приступності сегмента використано константи avail, bavail для звичайного і PBB сегмента відповідно.

На головній сторінці моделі зазначені C-MAC адреси абонентського устаткування в позиціях a11-a4c і B-MAC адреси PBB-комутаторів у позиціях ab0-ab5; MAC адреси представлені цілими числами, що не обмежує загальності, якщо розглядати, наприклад, тільки останній байт MAC адрес, що збігаються в перших 5-ти байтах.

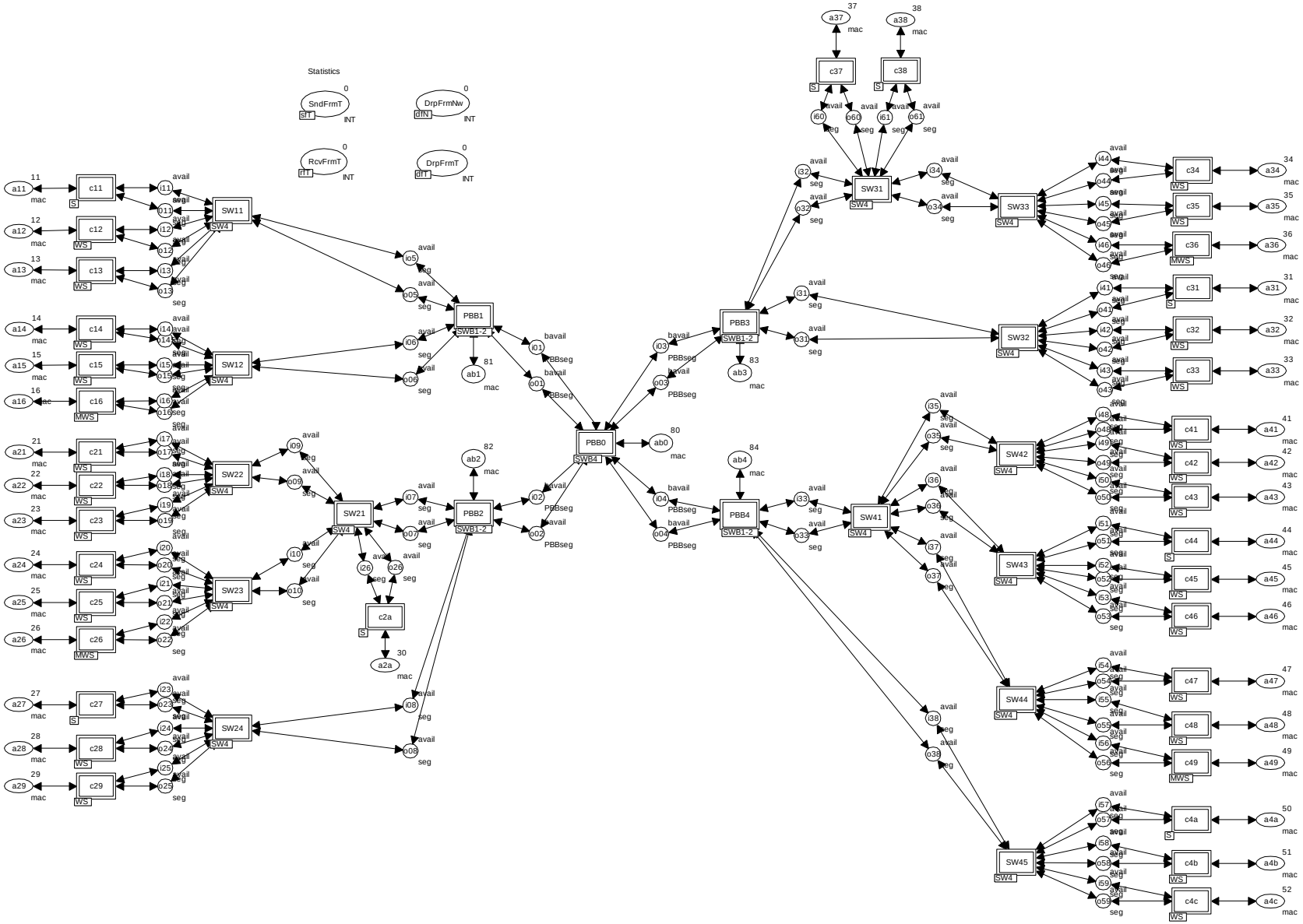


Рисунок 4.12 – Модель PBB мережі

Кожен порт комутаторів представлений парою позицій, що моделюють повнодуплексний режим роботи. Позиція ik – вхідний (input) канал k-го порту; позиція ok – вихідний (output). З'єднання устаткування у відповідності зі структурною схемою мережі виконується шляхом сполучення вхідних і вихідних позицій портів. Слід зазначити, що при з'єднанні двох комутаторів вхідний канал одного з них сполучається з вихідним каналом іншого і навпаки; назви позицій портів обрані відносно комутатора найбільш близького до PBB0.

Крім того, на головну сторінку винесені сполучені позиції для оцінки трафіка: SndFrm виду sf – лічильник відправлених кадрів, RcvFrm виду rf – лічильник прийнятих кадрів, DrpFrmNw виду df – лічильник загублених кадрів мережі, DrpFrm виду df – лічильник загублених кадрів термінального устаткування.

4.5.2 Аналіз результатів моделювання PBB мереж

Спочатку виконане роздільне налагодження компонентів, а потім комплексне налагодження моделі мережі; виконані трасування процесів доставки окремих кадрів і заповнення адресних таблиць. За допомогою додаткових лічильників установлено, що усі відправлені кадри доставляються за призначенням. Динамічно побудовані адресні таблиці цілком відповідають структурній схемі мережі.

Одиниця модельного часу (MTU) дорівнює 1,2 мкс, що відповідає часу передачі кадрів у 10Gbps сегменті. Час роботи компонентів устаткування не моделювався, тому що для цього вимагаються наносекундний масштаб часу, що утрудняє оцінки роботи мережі на тривалих часових інтервалах.

Вимірювальні фрагменти моделі представлені вимірювальними робочими станціями MWS для оцінки часу відгуку мережі і лічильниками кадрів для оцінки продуктивності і корисної продуктивності. Джерелом накладних витрат у технології Ethernet, включаючи PBB, є використання широкомовлення, а також витрати ресурсів на побудову покриваючих дерев. Дійсна модель дозволяє оцінити частку продуктивності мережі, затрачуваної на широкомовлення; оцінка роботи алгоритму покриваючого дерева не виконувалася.

Загальне число доставлених термінальному устаткуванню кадрів широкомовлення містить лічильник DrpFrm множини df на головній сторінці моделі; загальне число доставлених корисних кадрів – лічильник RcvFrm множини rf. При зупинці моделі за часом уміст RcvFrm менше вмісту SndFrm – лічильника відправлених кадрів, що зв'язано з тим, що певне число кадрів знаходиться в процесі доставки в мережі; однак, при генерації заданого числа кадрів і зупинці моделі по відсутності подій значення обох лічильників збігаються.

Корисна продуктивність мережі істотно залежить від часу старіння записів адресних таблиць TCL. Крім того, при включенні мережі (підключенні нових підмереж) створюється короткочасне перевантаження, викликане інтенсивним широкомовленням, що приводить до тимчасового зниження якості обслуговування (часу відгуку мережі). Динаміка широкомовлення і часу відгуку (як показника якості обслуговування QoS) представлені на Рис. 4.13. На графіках показаний сплеск широкомовлення при включенні мережі і його вплив на час відгуку, а також друга хвиля широкомовлення після очищення записів таблиць (через 12 с.), вплив якої згладжується в часі.

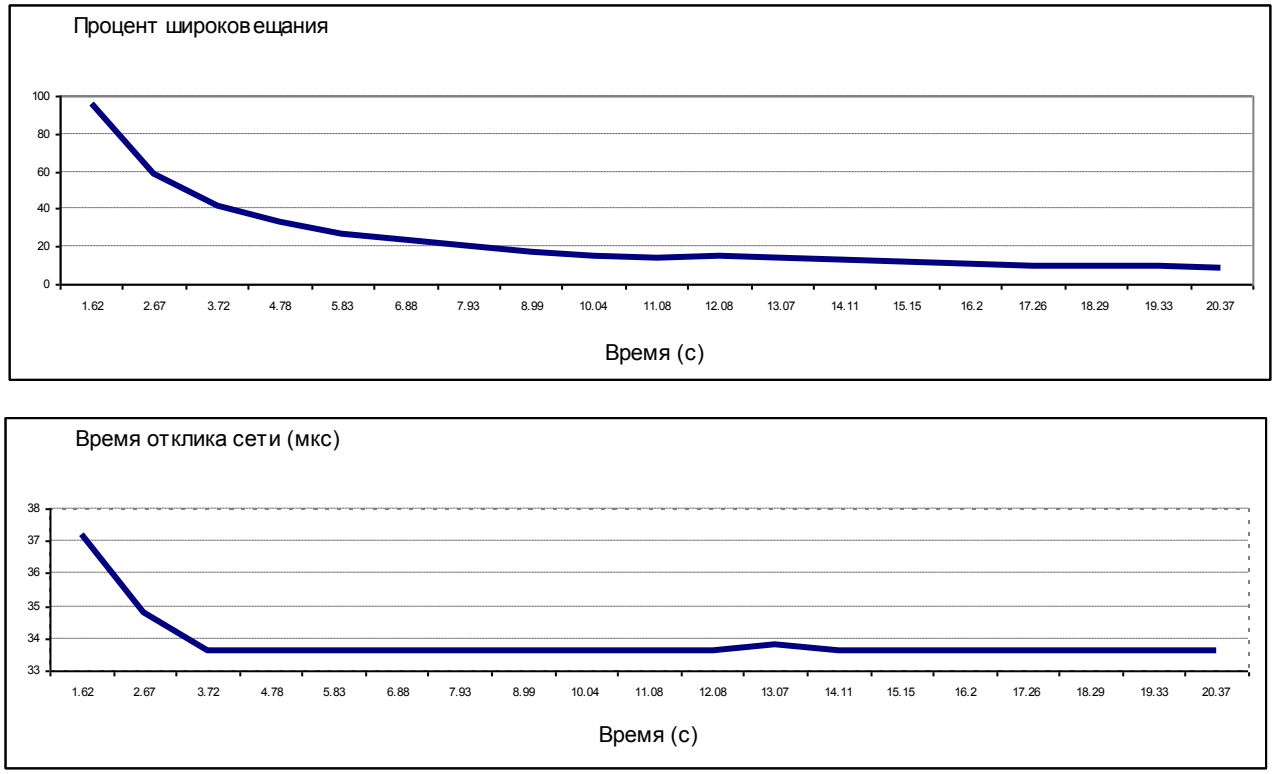


Рисунок 4.13 – Динаміка широкомовлення і QoS після включення

Залежність корисної продуктивності і часу відгуку мережі від часу старіння записів приведені на Рис. 4.14. Збільшення часу старіння записів приводить до підвищення продуктивності мережі і якості обслуговування, однак знижує можливості мережі по адаптації до зміни структури і приводить до неправильної доставки кадрів у результаті використання неактуальних записів адресних таблиць.

Вплив часу старіння записів підсилюється при збільшенні інтенсивності трафіка відповідно до оцінок, представленими на Рис. 4.15. Корисна продуктивність мережі підвищується при збільшенні інтенсивності трафіка (хоча і приводить до зниження якості обслуговування), що зв'язано зі збільшенням частоти використання записів адресних таблиць.

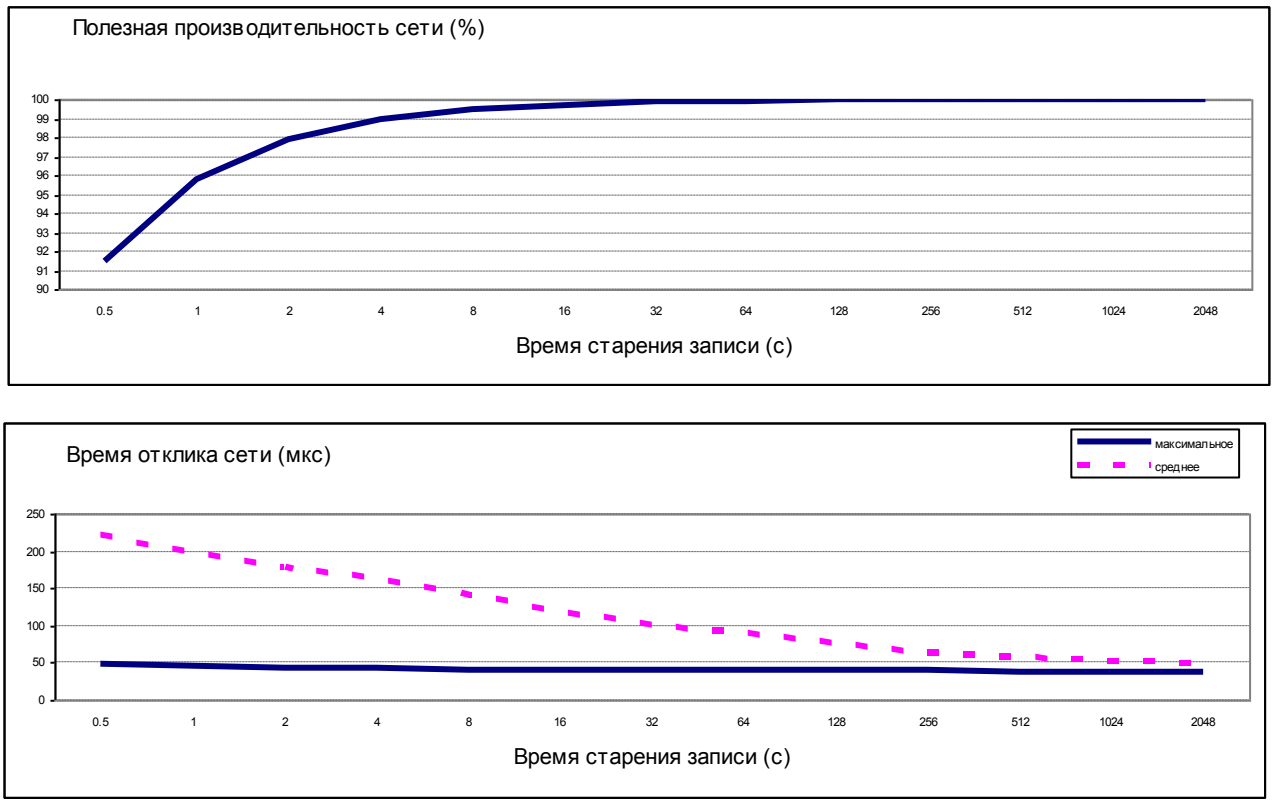


Рисунок 4.14 – Вплив часу старіння запису на продуктивність і QoS

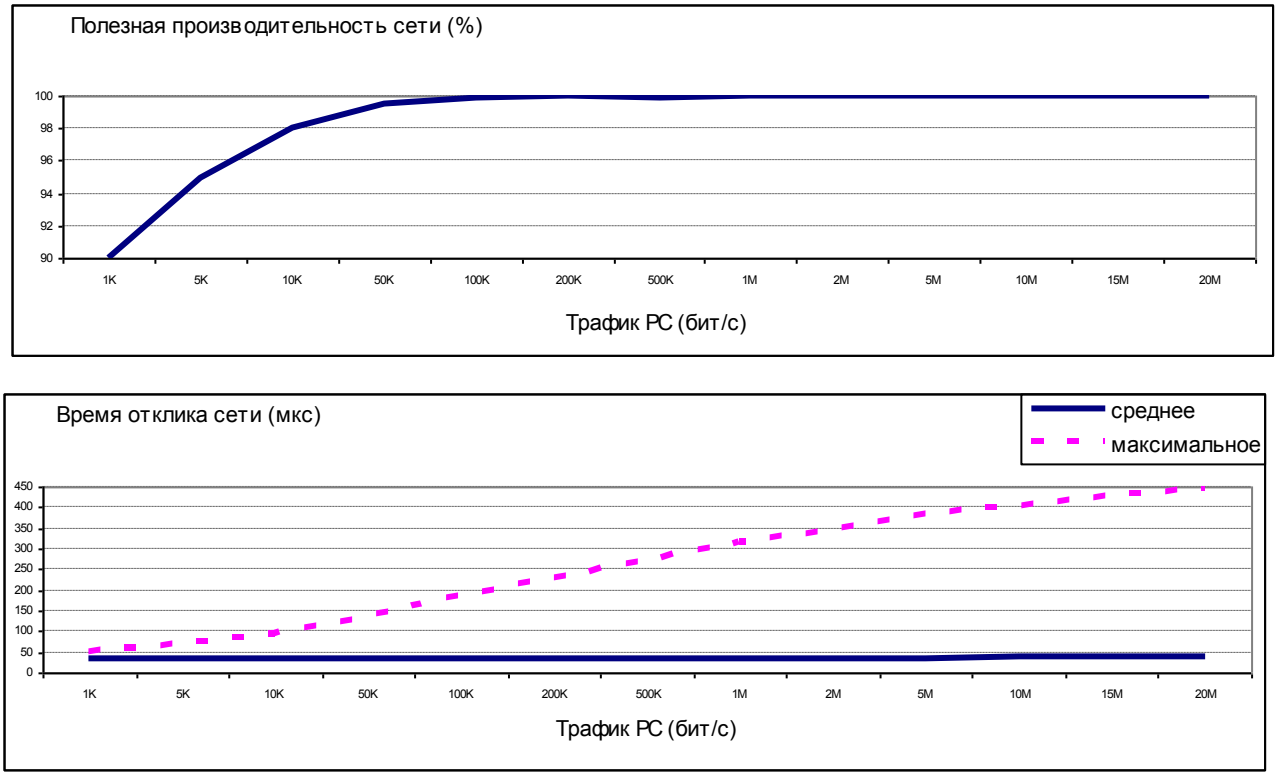


Рисунок 4.15 – Вплив інтенсивності трафіка на продуктивність і QoS

Таким чином, незважаючи на істотні переваги, технологія PBB має визначені недоліки, що утрудняють забезпечення заданої якості обслуговування, і потребує резервування продуктивності для згладжування можливих перевантажень. Слід зазначити, що в дійсній роботі не розглядалися віртуальні мережі, що дозволяють ізолювати широкомовний трафік (у межах віртуальної мережі).

На основі моделей, побудованих у дійсному розділі, і моделей E6 мереж Розділу 3 можна виконати попередній порівняльний аналіз двох технологій. Відсутність службового широкомовлення в E6 дозволяє забезпечити гарантовану якість обслуговування. Ієрархічна структура E6 мережі (включаючи магістраль) задається за допомогою ієрархічних E6 адрес і не приводить до збільшення заголовка кадру; у PBB кожен новий рівень ієрархії магістралі вимагає не менш 12 додаткових байтів заголовка кадру (для пари B-MAC адрес). Крім того, стек E6 анулює 20-36 байтів заголовків протоколів TCP, IP кожного інкапсульованого пакета. Технологія PBB при інкапсуляції IP-Ethernet вимагає подвійне відображення адрес: IP->C-MAC, C-MAC->B-MAC,– у той час як E6 цілком анулює відображення адрес у мережі за рахунок використання єдиної E6 адреси.

Висновки до 4 розділу

1. Виконано аналіз технології магістральних мостів провайдера PBB стосовно доставки кадрів та використання адресних таблиць для чого побудовано загальну модель комутатору з динамічним поновленням адресних таблиць, а також моделі чотирьох різновидів портів комутатора: традиційного, магістрального PBB, C-порту та B-порту граничного PBB.

2. На основі аналізу результатів моделювання PBB мереж знайдено певні недоліки PBB технології, обумовлених чутливістю до часу старіння записів адресних таблиць та широкомовленням, що утрудняють забезпечення заданої якості обслуговування. Виконані порівняння підтверджують ряд істотних переваг E6 адресації відносно до технології PBB.

5 РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ Е6 МЕРЕЖ

В попередніх розділах виконане моделювання Е6 мереж у середовищі моделюючої системи CPN Tools [16,35]. Побудовано моделі робочих станцій, серверів, термінальних (абонентських) Е6 мереж для генерації потокового трафика, а, саме головне, – модель комутуючого маршрутизатора Е6 (КМЕ6). Перераховані компоненти є основою для побудови моделей і дослідження Е6 мереж заданих їхніми структурними схемами. Дослідження серії мереж підтвердили не тільки працездатність, але також ефективність технології Е6. Однак дослідження початкового етапу розробки Е6 технології [7,8] абстрагувалися від оточення існуючих мережних технологій, у якому можлива практична реалізація технології Е6.

Метою дійсного розділу є дослідження питань роботи стека протоколів Е6 у реальному мережному середовищі в оточенні інших мережних технологій [24,25,31] і формування принципів побудови шлюзів Е6 й TCP/IP мереж, а також експериментальна програмна реалізація стеку протоколів Е6 [11].

5.1 Стратегія впровадження Е6 мереж

При попереднім обговоренні переваг технології Е6 [7,8] малася на увазі стратегія *повного заміщення* технології TCP/IP (Розділ 2). Однак застосування такої стратегії навіть при повнім збереженні прикладного програмного забезпечення не представляється реалістичним. Потрібне одноразова зміна мережного середовища в досить великих масштабах, що не представляється практично здійсненним, за винятком досить невеликих корпорацій. Крім того, більшість сучасних операційних систем, наприклад Unix (Linux) [2,18,28,48], у значній мірі інтегровані зі стеком протоколів TCP/IP, що утрудняє повне заміщення й вимагає перекомпіляції всіх використовуваних операційних систем.

На основі реалістичної оцінки сучасного стану й перспектив розвитку телекомунікаційних мереж і їхніх операційних середовищ запропонована стратегія *поступового витиснення* технології TCP/IP при впровадженні технології Е6. Для забезпечення витиснення необхідна паралельна (стосовно стеку TCP/IP) реалізація стека Е6. При цьому драйвери мережних карт Ethernet, що утворять канальний рівень стека, можуть бути використані в практично незмінному вигляді. Стандартно Ethernet драйвер виконує мультиплексування / демультиплексування кадрів на основі поля типу кадру Ethernet. Наприклад, пакету IP відповідає значення 0x0800, пакету Appletalk – 0x809B [67]. Запропоновано стандартизувати

додатково тип E6 кадру, наприклад, зі значенням поля типу, рівним 0xe600. Рис. 5.1 ілюструє принципи незалежної спільної роботи стеків протоколів E6 і TCP/IP.

При виклику драйвера Ethernet стандартно передається номер протоколу, що потім розміщається в полі типу кадру. Модуль E6-Узгодження викликає драйвер, указуючи номер протоколу 0xe600, а модуль IP - указуючи номер протоколу 0x0800. Комутатори Ethernet у стандартній конфігурації не інтерпретують тип кадру (за винятком декількох спеціальних типів). Розглядаючи адресу призначення кадру як ключ пошуку в адресній таблиці комутатор визначає порт для перенапрямку кадру; при відсутності адреси в таблиці виконується широкомовлення (ретрансляція прийнятого кадру по всіх портах крім вхідного). Для заповнення адресних таблиць комутатор використає пасивне прослуховування адрес відправника кадрів.

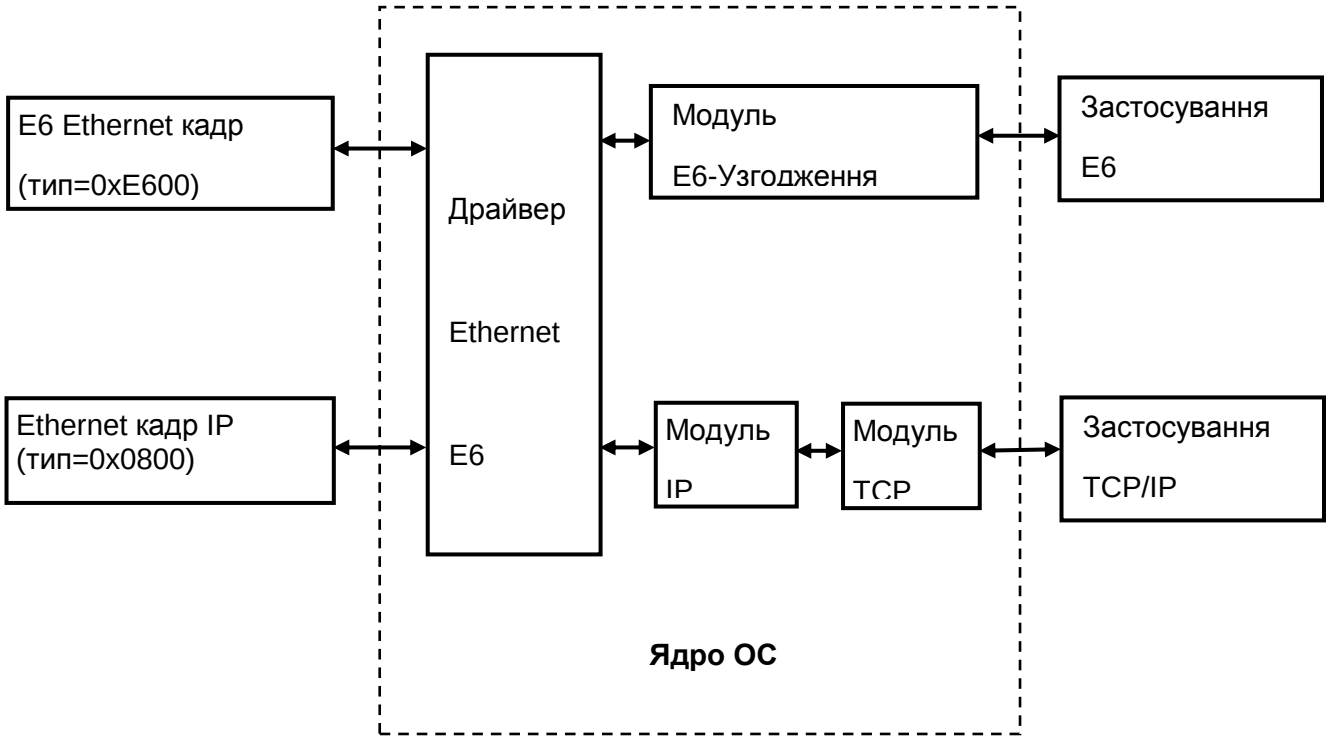


Рисунок 5.1 – Спільна робота стеків E6 і TCP/IP

Помітимо, що комутатор не інтерпретує MAC адреси кадру (за винятком перших двох бітів). Стандартна структура MAC адреси, що складається з коду компанії-виробника й індивідуального номера пристрою, не інтерпретується комутатором. Таким чином, вказівка E6 адреси замість MAC адреси не відіб'ється на нормальній роботі комутатора. Всі E6 кадри будуть доставлені по призначенню в межах комутованої Ethernet.

Побічним ефектом передачі Е6 кадрів у комутованій Ethernet є їхня доставка в результаті широкомовлення хостам, не підтримуючим стік Е6. Така доставка не впливає на нормальне функціонування пристроїв у відповідності зі стандартами. Всі доставлені пакети з невідомим типом кадру (номером протоколу) будуть знищені (загублені) приймаючим пристроєм.

Тому як *перший етап* впровадження технології Е6 пропонується реалізація стека протоколів Е6 на робочих станціях і серверах для забезпечення експлуатації в межах комутованої мережі Ethernet. Помітимо, що навіть при збільшенні розмірів комутованої Ethernet, переваги технології Е6 не будуть розкриті повною мірою. У стандартній комутованій Ethernet Е6 адреса використається точно також як Ethernet MAC адреса. А саме: кожна індивідуальна адреса перераховується в адресній таблиці. Розмір таблиць повинен бути порівнянним із загальною кількістю абонентських пристроїв у мережі, що практично нездійсненно в глобальних мережах.

Другий етап впровадження припускає створення специфічних для технології Е6 засобів доставки Е6 пакетів (кадрів). Відповідно до [7,8] основним мережним пристроєм є комутуючий маршрутизатор Е6 (КМЕ6). На відміну від комутатора Ethernet він забезпечує агрегування індивідуальних Е6 адрес під загальною маскою й у такий спосіб істотно скорочує число записів адресних таблиць, дозволяючи будувати всесвітні мережі.

Незважаючи на подібність із IP мережами, що також використають ієрархічні адреси, Е6 мережі мають ряд істотних переваг: Е6 адреса на два байти довше від IP адреси, що забезпечує розширення простору адрес в 16 тис. раз.; відсутнє відображення адрес, що знижує час обробки пакета; Е6 адреса використається на всіх рівнях еталонної моделі, включаючи фізичний, що ліквідує повторну інкапсуляцію / деінкапсуляцію пакету на кожному проміжному маршрутизаторі. Зазначені переваги ілюструє порівняння схем доставки Е6 і IP пакетів, представлених на Рис. 1.5, 1.6. Певні числові показники порівняння здобуто в Розділах 2, 3.

При реалізації КМЕ6 можна використати або програмний, або програмно-апаратний підходи [1,25]. Програмний підхід припускає використання комп'ютера загального призначення, що має декілька Ethernet інтерфейсів (карт), у якості КМЕ6. Програмне забезпечення Е6 маршрутизації варто включити в комплекс програм стека Е6. Недоліком такого підходу є порівняно низька швидкість перемикання пакетів.

Полярним підходом є апаратна реалізація КМЕ6, що забезпечить максимально можливу продуктивність мережі за рахунок оптимізації не тільки програмного забезпечення, але також архітектури апаратних засобів. Відповідні оцінки покращення продуктивності та якості обслуговування отримано в Розділі 2. Однак подібний підхід вимагає промислових інвестицій.

Досить прийнятним варіантом є розробка модулів програмного забезпечення маршрутизації Е6 для існуючих маршрутизаторів, наприклад для операційної системи IOS

компанії CISCO [1]. Однак такий підхід вимагає спеціальних ліцензійних угод з відомими фірмами-виробниками.

Помітимо, що навіть при використанні тих самих пристроїв у якості IP маршрутизатора й КМЕ6, доставка Е6 і IP пакетів залишається незалежної відповідно до маршрутних таблиць кожного зі стеків. Таким чином, створюються умови для паралельного розвитку мереж технології Е6.

Слід відзначити, що існуючі в дійсний час значні інформаційні ресурси Інтернет (TCP/IP мереж) будуть сприяти в багатьох випадках вибору TCP/IP навіть незважаючи не більше високу ефективність Е6. Таким чином, створення шлюзу між Е6 і TCP/IP мережами є ключовим рішенням, що дозволяє створити рівні умови для вільної конкуренції двох технологій, що і являє собою зміст *третього етапу* впровадження Е6 мереж.

5.2 Організація шлюзів Е6 та TCP/IP мереж

Основною вимогою до організації шлюзу мереж повинне бути забезпечення максимальне можливої зручності для кінцевого користувача. З погляду кінцевого користувача адресація є внутрішнім механізмом мережі, оскільки для ідентифікації ресурсів мережі використаються єдині локатори ресурсів (URL) [36,71], у яких хост вказується за допомогою доменного імені. Реалізація Е6-DNS являє собою по суті перекомпіляцію системи імен IP мереж DNS [60,61] з розширенням поля адреси з 4 до 6 байтів.

Таким чином, вихідним станом для створення шлюзу є паралельно реалізована Е6 мережа із власною системою доменних імен Е6-DNS. Цілком обґрунтовано представляти власну систему доменних імен без спеціальних суфіксів, а використати додатковий суфікс для ідентифікації хоста (ресурсу) чужої мережі: суфікс «.e6» для вказівки імені Е6 мережі усередині IP мережі; суфікс «.ip» для вказівки імені IP мережі усередині Е6 мережі. Наприклад, IP хост onat.edu.ua при вказівці з Е6 мережі буде пойменований як onat.edu.ua.ip і навпаки, Е6 хост geocities.com при вказівці з IP мережі буде пойменований як geocities.com.e6.

Пропонується організувати шлюз як спеціальне прикладне програмне забезпечення, що функціонує на основі принципів трансляції адрес [42] на комп'ютерах, що підтримують як стек протоколів Е6, так і стек протоколів IP. Помітимо, що наявність двох відповідних різних фізичних інтерфейсів не є обов'язковою. Шлюзів може бути значна кількість для забезпечення ефективності при виборі маршруту. Для визначення адреси найближчого шлюзу може бути використаний авторизований (E6-)DNS поточної зони. Адреса шлюзу повинна бути відомою прикладному програмному забезпеченню (браузеру); вона є адресою шлюзу того ж самого типу,

що й поточна мережа: E6 адреса для E6 мережі, IP адреса для IP мережі. Схема організації шлюзу наведена на Рис. 5.2.

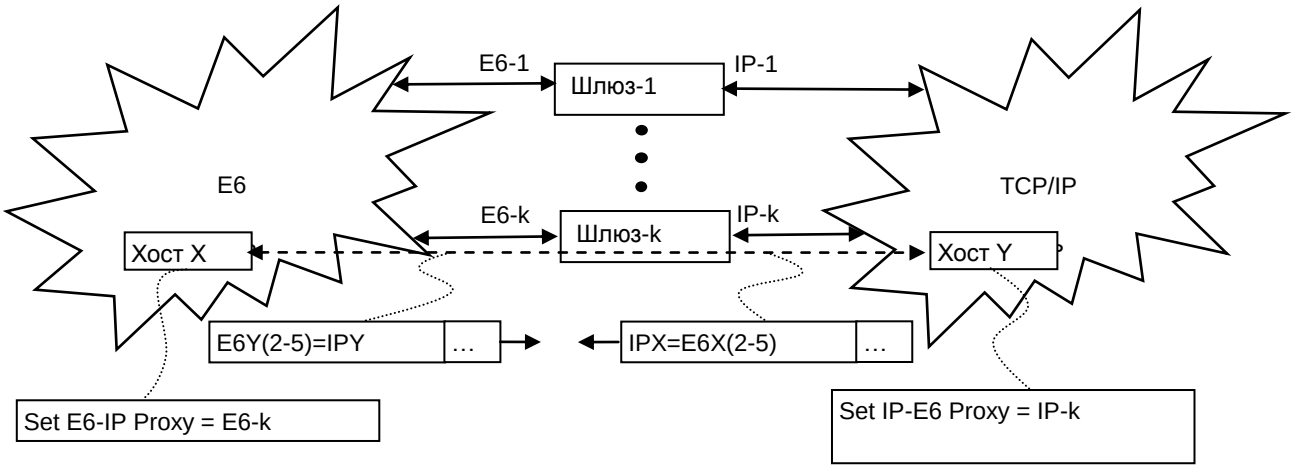


Рисунок 5.2 – Схема організації шлюзу E6-IP

На відміну від посередників трансляції адрес в IP мережах (proxy NAT IP), шлюз E6-IP повинен вирішити проблеми, пов'язані з різною довжиною E6 і IP адреси. Причому найбільш складним є питання організації доступу до ресурсів E6 мереж з IP мереж, оскільки не передбачається модифікація програмного забезпечення IP мережі. Для загальної організації роботи шлюзу резервується певний IP і E6 порт, наприклад 230.

Розглянемо використання шлюзу в E6 мережі. У програмному забезпеченні необхідно встановити E6 адресу шлюзу E6-k. Для передачі IP адреси використати E6 адресу із префіксом 1.0. - ознакою IP мережі. При вказівці E6 адреси призначення із префіксом 1.0. переадресовувати пакети на шлюз E6-k.

Розглянемо використання шлюзу в IP мережі. Адреса шлюзу IP-k може бути зазначена, наприклад у браузері, як адреса стандартного proxy-сервера. Таким чином, протокол роботи шлюзу з боку IP мережі повинен повністю відповідати протоколу proxy-IP. Для передачі молодших 4 байтів E6 адреси запропоновано використати IP адресу. Для вказівки старших двох байтів увести додаткову команду proxy-сервера Set_e6_Prefix - установити префікс E6 адреси (перші два байти). Додаткова команда може генеруватися першим пакетом, спрямованим до шлюзу, що випереджає стандартні пакети; для реалізації зазначеної технології необхідна модифікація браузера (патч).

Найбільш витонченим представляється спільна реалізація шлюзу й E6-DNS сервера IP мережі, що для імен із суфіксом «.e6», запитаних з IP мережі повертає молодші 4 байти E6 адреси як IP адресу й запам'ятовує поточний префікс для IPY, з якого була запитана E6 адреса, для наступного формування E6 адреси. У цьому випадку не потрібна модифікація браузера IP

мережі.

Альтернативним підходом до використання префіксів є реалізація імен незалежних від обраної технології із забезпеченням можливості міграції імені між TCP/IP та E6 мережами прихованої від користувача. Тоді E6-DNS таблиці доповнюються ознакою технології мережі; таблиця існуючої TCP/IP DNS [61] переадресує запит на один із шлюзів.

5.3 Принципи програмної реалізації стека E6

Реалізація стека протоколів E6 припускає інтеграцію з операційним середовищем комп'ютера, представленим певною операційною системою [18,28]. Як правило ОС складається з ядра, що працює в захищеному режимі процесора і системних процесів [2]. Мережні і транспортні протоколи вимагають тісної інтеграції з драйверами пристроїв і повинні забезпечувати високу продуктивність при передачі повідомлень і сегментів інформації. Тому традиційно протоколи рівнів OSI-ISO від подання інформації до канального реалізують у ядрі ОС [48]. Таким чином, стек певного сімейства протоколів представлений набором взаємодіючих процедур ядра ОС і відповідних структур даних [4]. Зазначена структура як правило локалізується в рамках досить автономного модуля, що може бути включений у ядро в процесі його компіляції і компонування [18,38]. Модуль має три основних інтерфейси: інтерфейс застосувань, інтерфейс фізичного устаткування, інтерфейс середовища ядра ОС.

Основним проектним рішенням експериментальної реалізації був вибір одного з двох шляхів: заміщення модулів існуючого стеку протоколів TCP/IP стеком E6; додання модулів стеку E6. Порівняння двох стеків наведено на Рис. 1.4. Заміщення пов’язано з повною модифікацією майже всіх компонентів ОС і є занадто трудомістким. Тому саме обрано шлях додання модулів стеку E6 та забезпечення їх сумісного функціонування поруч з існуючими модулями стеку TCP/IP. Для цього було запропоновано використати нове значення типу кадрів E6Ethernet (0xE800) для позначення кадрів E6 та забезпечення їх успішного мультиплексування/демультиплексування драйвером Ethernet поруч з кадрами інших існуючих протоколів.

Оскільки передбачається перенос стека E6 у різні операційні середовища: UNIX, QNX, MS WINDOWS та інші [28], необхідно забезпечити мінімально можливу залежність реалізації від інтерфейсу середовища ядра ОС. Тому обраний підхід безпосередньої реалізації процедур інтерфейсу застосувань протоколів UDP та TCP, перерахованих у стандартах IETF [63,65]: SEND, RECEIVE, CREATE_PORT для UDP; OPEN, SEND, RECEIVE, CLOSE, ABORT, STATUS для TCP. При створенні зазначених процедур до їхнього стандартного імені доданий префікс E6.

Отже обрано наступні принципи програмної реалізації E6:

- незалежно від існуючих стеків;
- забезпечення інтерфейсів з прикладним рівнем безпосередньо за стандартами IETF;
- подальше створення шлюзів з існуючими мережними технологіями.

5.4 Експериментальна реалізація стека E6 у ядрі ОС Linux

Операційні системи Linux мають відкритий вихідний код і містять інструментальні засоби для реалізації нових стеків протоколів [18,38], що обумовило їхній вибір як операційне середовище. Основним принципом реалізації був вибраний підхід збереження прикладних інтерфейсів у відповідності зі стандартами TCP і UDP [63,65] і канального інтерфейсу у відповідності зі стандартами Ethernet [51-55], а також забезпечення незалежної роботи стека E6 серед інших протоколів.

Дійсна експериментальна реалізація виконана у ядрі версії 2.6.22.9 та не використає засоби Ethernet LLC2 [51] і забезпечує прикладний інтерфейс протоколу UPD; крім того організовано роботу лише з одним інтерфейсом Ethernet без перенаправлення пакетів. У подальшому описі наведено основні фрагменти програм на мові C [17] адаптовані для вивчення; повний текст програм (версія без буферування повідомлень) наведено у Додатку А.

5.4.1 Прикладні інтерфейси

Програмне забезпечення розроблене як завантажуваний модуль e6udpmod.ko ядра Linux [2,38,48] за винятком незначних змін внесених у статичну частину ядра з метою додавання нового системного виклику. Умовний (conditional) системний виклик 324 з ім'ям e6_call доданий у статичну частину ядра. Цей додатковий системний виклик призначений для реалізації всіх дійсних і майбутніх процедур прикладного інтерфейсу стека E6. Статична частина ядра містить покажчик для нового системного виклику 324 (який не задіяний у версії ядра 2.6.22.9) на фіктивну процедуру sys_e6call, що перевіряє покажчик new_sys_e6call на дійсну процедуру й викликає її, у випадку якщо покажчик ненульовий (не дорівнює NULL). Покажчик ініціюється значенням NULL так що коли модуль e6updmod.ko не завантажений нічого не відбувається за винятком того, що sys_e6call може виводити повідомлення в системний журнал для індикації виклику. Покажчик new_sys_e6call експортується ядром для використання у завантажуваних модулях. Програмний код має наступний вигляд:

```
/* file syscall_table.S */
ENTRY(sys_call_table)
....
      .long sys_e6call      /* new e6 syscall 324 */

/* file socket.c  e6 new */
long (*new_sys_e6call)(int call, unsigned long __user *args) = NULL;
asmlinkage long sys_e6call(int call, unsigned long __user *args)
{
  int err;
  if( new_sys_e6call != NULL )
    err = (*new_sys_e6call)( call, args );
  else
    { printk(KERN_INFO"*** sys_e6call echo: %d\n", call); err=0; }
  return err;
}
EXPORT_SYMBOL(new_sys_e6call);
/* e6 new end */
```

Усі інші дії виконує модуль e6udpmod.ko. При ініціалізації він установлює покажчик new_sys_e6call на дійсну точку входу оброблювача системних викликів E6, а також знаходить Ethernet пристрій e6_dev серед зареєстрованих пристроїв ядра, копіює його апаратну адресу й реєструє оброблювач нових E6 пакетів:

```
static int e6udpmod_init_module(void)
{
      int err=0;

      printk(KERN_INFO"*** e6udpmod: init devname=%s ***\n", e6_devname);
      /* find e6 device */
      e6_dev = dev_get_by_name(e6_devname);
      memcpy( e6_myaddr, e6_dev->dev_addr, e6_dev->addr_len );
      /* set new e6call function */
      new_sys_e6call = mod_sys_e6call;
      /* register e6 packet handler */
      dev_add_pack(&e6_packet_type);

         return 0;
}
```

Після оператора new_sys_e6call = mod_sys_e6call всі системні виклики E6 обробляються підпрограмою mod_sys_e6call, що розташована в модулі e6updmod.ko і працює як диспетчер умовних системних викликів розпізнаваних по їхніх номерах, заданим змінною call:

```
extern long (*new_sys_e6call)(int call, unsigned long __user *args);

long mod_sys_e6call(int call, unsigned long __user *args)
{
  unsigned long a[1];
  unsigned long a0;
  unsigned char nargs = (1)*sizeof( unsigned long );
  int err;

  printk(KERN_INFO"*** e6udpmod: mod_sys_e6call %d\n", call );

  if (copy_from_user(a, args, nargs))
      return E6ERR_COPY;

  a0 = a[0];

  switch( call ){
  case E6_CALL_PUTMSG:
           err = e6_putmsg( (struct e6msg_buf __user *)a0 );
           break;
  case E6_CALL_GETMSG:
           err = e6_getmsg( (struct e6msg_buf __user *)a0 );
           break;
...
  default:
           err = E6ERR_CALLNUM;
           break;
  }

  return err;
}
```

Інтерфейс на прикладному рівні забезпечується бібліотекою e6udplib, що містить функції кінцевого користувача e6sendmsg, e6rcvmsg, e6regport, e6unregport, e6waitmsg, які видають у результаті своєї роботи системний виклик 324, використовуючи процедуру syscall зі стандартної бібліотеки libc:

```
#define SYS_e6call         324
#define E6_CALL_PUTMSG     1
#define E6_CALL_GETMSG     2
...
#define MAXE6BUFSIZE       1496
#define E6ADDRLEN          6

struct e6msg_buf {
     u8 e6dst_addr[E6ADDRLEN];
     u8 e6src_addr[E6ADDRLEN];
     u16 e6dst_port;
     u16 e6src_port;
     u16 e6data_len;
     u8 * e6data;
};

struct e6msg_buf buf;

int e6sendmsg( struct e6msg_buf * b, int * err )
{
 int rc;
 unsigned long a[1];

 a[0]=(unsigned long)b;

 rc = syscall( SYS_e6call, E6_CALL_PUTMSG, a );

 *err = errno;

 return rc;
}
...
```

Множина програм e6sendmsg, e6rcvmsg, e6regport, e6unregport, e6waitmsg повністю задовольняє вимогам RFC 768 до прикладних інтерфейсів протоколу UDP. Взаємодія частин програмного коду, розміщених у статичній області ядра, у завантажуваному модулі і в користувальницькій бібліотеці, проілюстровано на Рис. 5.3.

Бібліотека інтерфейсу користувача e6udplib задіяна для розробки простого застосування e6talk який забезпечує обмін текстовими повідомленнями в межах E6 мережі, побудованої на основі комутованої Ethernet. Зазначене застосування виконує реальний обмін інформацією, що

підтверджує працездатність стека протоколів E6. Для підтвердження ефективності технології E6 відносно сімейства протоколів TCP/IP необхідно ще виконати досить велику роботу.

Аналізатори трафіку, такі як Wireshark, дозволяють спостерігати E6 Ethernet кадри, які вільно передаються серед інших типів кадрів і не перешкоджають роботі інших протоколів (Додаток Б). Таким чином, E6 мережі існують у паралельному світі стосовно TCP/IP доти поки не будуть розроблені шлюзи між E6 і TCP/IP мережам, що є напрямком майбутніх робіт.

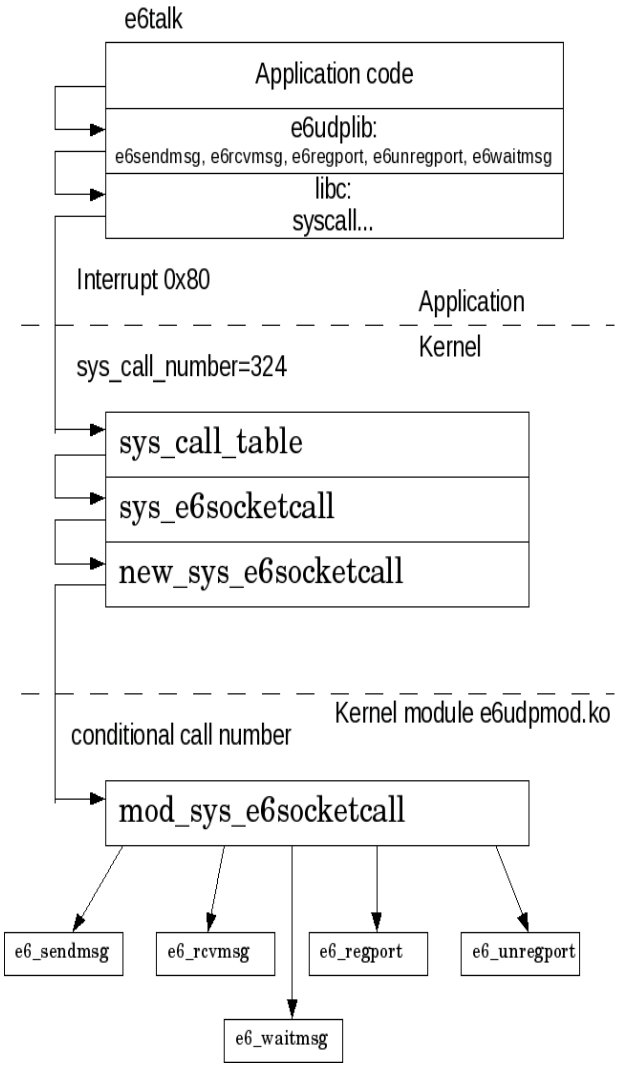


Рисунок 5.3 – Взаємодія між програмним кодом користувача, ядра й модуля

5.4.2 Інтерфейси канального рівня

Інтерфейс із устаткуванням Ethernet забезпечується в середовищі ядра Linux структурою struct net_device, що містить опис пристрою і його функцій. Припустимі функції задані набором покажчиків, які вказують на дійсні програми поточного драйвера Ethernet; основними програмами є наступні: open, stop, hard_start_xmit. Структура net_device містить також

заголовок черги вихідних пакетів; пакет представлений описом struct sk_buff *skb.

Інтерфейси канального рівня залишені без змін; вони тільки використані для передачі E6 Ethernet кадрів. Мультиплексування при відправленні E6 пакета здійснюється за допомогою використання нового типу E6 Ethernet кадрів:

```
#define ETH_P_E6      0xe600
```

Модуль заповнює sk_buff даними користувача й E6 заголовками, використовуючи тип кадру ETH_P_E6, і викликає програму hard_start_xmit, передаючи керування драйверу Ethernet для передачі пакета через середовище.

Одержання E6 пакета є більше складним, тому що воно виконується драйвером Ethernet по апаратному перериванню Ethernet адаптера (карти). Драйвер виділяє пам'ять і заповнює sk_buff даними прийнятого кадру й потім виконує процедуру демультиплексування, засновану на множині зареєстрованих типів пакетів. Реєстрація нового типу пакета здійснюється за допомогою структури struct packet_type; структура ініційована в такий спосіб:

```
static struct packet_type e6_packet_type = {
      .type = __constant_htons(ETH_P_E6),
      .func = e6_rcv,
      .gso_send_check = NULL,     /*e6_gso_send_check,*/
      .gso_segment = NULL,        /*e6_gso_segment,*/
};
```

Таким чином, після наступного оператора в зазначеній вище процедурі ініціалізації модуля (e6udpmod_init_module)

```
dev_add_pack(&e6_packet_type);
```

всі прийняті пакети типу ETH_P_E6 обробляються програмою e6_rcv розташованої усередині модуля e6udpmod.ko. Отже, установлені двосторонні інтерфейси з канальним рівнем: для відправлення E6 пакетів і для одержання E6 пакетів. Зауважимо, що E6 пакети й відповідні E6 кадри передаються незалежно серед інших типів кадрів Ethernet. Схема, представлена на Рис. 5.4, пояснює інтерфейси модуля e6udpmod.ko з канальним рівнем Ethernet.

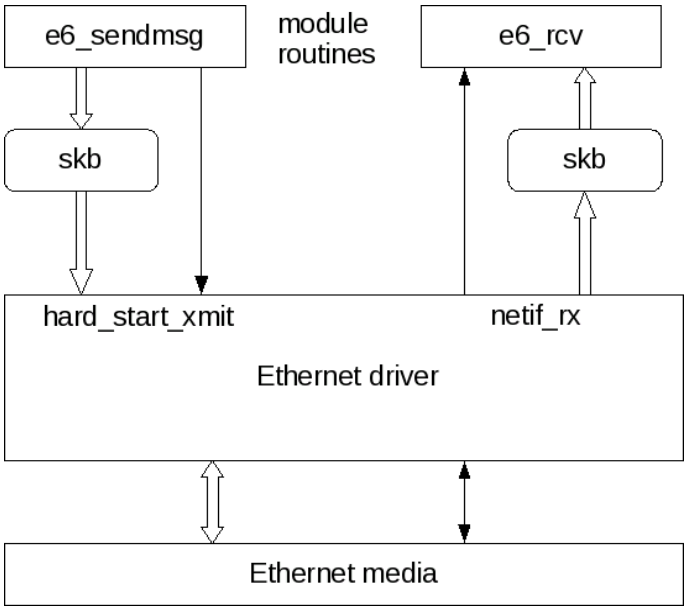


Рисунок 5.4 – Інтерфейси з канальним рівнем

5.4.3 Внутрішні структури даних і програми

Крім описаних вище програм ініціалізації модуля e6udpmod_init_module і диспетчера умовних системних викликів 324 з ім'ям mod_sys_e6call, модуль e6udpmod.ko містить наступні програми: процедури реалізації умовних викликів e6_sendmsg, e6_rcvmsg, e6_regport, e6_unregport, e6_waitmsg; програму для обробки прибулих E6 пакетів e6_rcv; програму виходу з модуля (деініціалізації) e6udpmod_cleanup_module. Взаємодія зазначених програм представлено на Рис. 5.3.

Розглянемо основні структури даних модуля e6udpmod.ko. Помітимо, що відповідно до RFC 768 порт джерела певного застосування повинен бути зареєстрований. Повідомлення може бути відправлено тільки із зареєстрованого порту також як повідомлення може бути отримано тільки на зареєстрований порт. Таким чином, основною структурою даних є список зареєстрованих портів (E6 сокетів). Через те що відправлення повідомлення повністю реалізовано протягом виконання одного системного виклику e6sendmsg, черги вихідних пакетів не створюються усередині модуля e6udpmod.ko; виділяється буфер пакета sk_buff, дані копіюються в буфер з користувальницького адресного простору, формуються E6 заголовки й, нарешті, sk_buff розміщається в черзі вихідних пакетів за допомогою виклику функції драйвера hard_start_xmit.

При одержанні нового E6 пакета програма e6_rcv викликається драйвером і пакет розміщається у відповідну чергу до зареєстрованого порту. Програма e6_rcv розміщає пакет у хвіст черги, системний виклик e6_rcvmsg витягає пакет з голови черги (дисципліна FIFO). Якщо порт призначення не зареєстрований (невідомий) пакет губиться (знищується).

Таким чином, створюються два рівні списків: список зареєстрованих портів і списки (черги) отриманих пакетів (до зареєстрованих портів). Рис. 5.5 ілюструє взаємозв'язок між основними структурами даних. Опис відповідних структур даних задано наступним програмним кодом:

```
struct e6portq {
      struct e6portq * next;
      struct e6portq * prev;
      u16 e6reg_port;
      pid_t pid;
      int qlen;
      struct sk_buf *skbhead;
      struct sk_buf *skbtail;
}

static struct e6portq * e6inpq_head = NULL;
static struct e6portq * e6inpq_tail = NULL;
```

Розглянемо детально процес відправлення E6 повідомлень за допомогою наступної програми:

```
int e6_sendmsg( struct e6msg_buf __user *msg )
{
      int len;
      struct sk_buff *skb;
      struct e6hdr *e6h;
      struct ethhdr *eth;
      unsigned long flags;
      int rc=0;

      copy_from_user(km, msg, sizeof(struct e6msg_buf)));
      len=km->len;
      if( e6find_regport( e6src_port ) == NULL )
      {
            rc = E6ERRUNREGPORT;
            return rc;
      }
      skb=alloc_skb(len+E6HEADSSPACE, GFP_KERNEL);
```

```
    skb->dev=e6_dev;
    skb->sk=NULL;

    /* Copy data */
    skb_reserve(skb, E6HEADSSPACE);
    copy_from_user( skb_put(skb,len), km->e6data, len );

    /* Build the E6P2 header. */
    skb_push(skb, sizeof(struct e6hdr));
    skb_reset_transport_header(skb);
    e6h = (struct e6hdr *)skb_transport_header(skb);
    e6h->e6dst_port = htons( km->e6dst_port );
    e6h->e6src_port = htons( km->e6src_port );
    skb_reset_network_header(skb);

    /* Build the E6Ethernet header */
    skb_push(skb, ETH_HLEN);
    skb_reset_mac_header(skb);
    eth = (struct ethhdr *)skb_mac_header(skb);
    skb->protocol = eth->h_proto = htons(ETH_P_E6);
    memcpy(eth->h_source, e6_myaddr, dev->addr_len);
    memcpy(eth->h_dest, km->e6dst_addr, dev->addr_len);

    /* Pass skb to the driver */
    local_irq_save(flags);
    netif_tx_lock(dev);
    rc=dev->hard_start_xmit(skb,dev);
    netif_tx_unlock(dev);
    local_irq_restore(flags);

    return rc;
}
```

Програма копіює заголовок з адресного простору користувача, перевіряє чи був зареєстрований порт джерела, виділяє буфер пакета skb, копіює дані з адресного простору користувача, заповнює заголовки e6h і eth і передає skb у драйвер. Розглянемо короткий опис алгоритмів інших програм модуля:

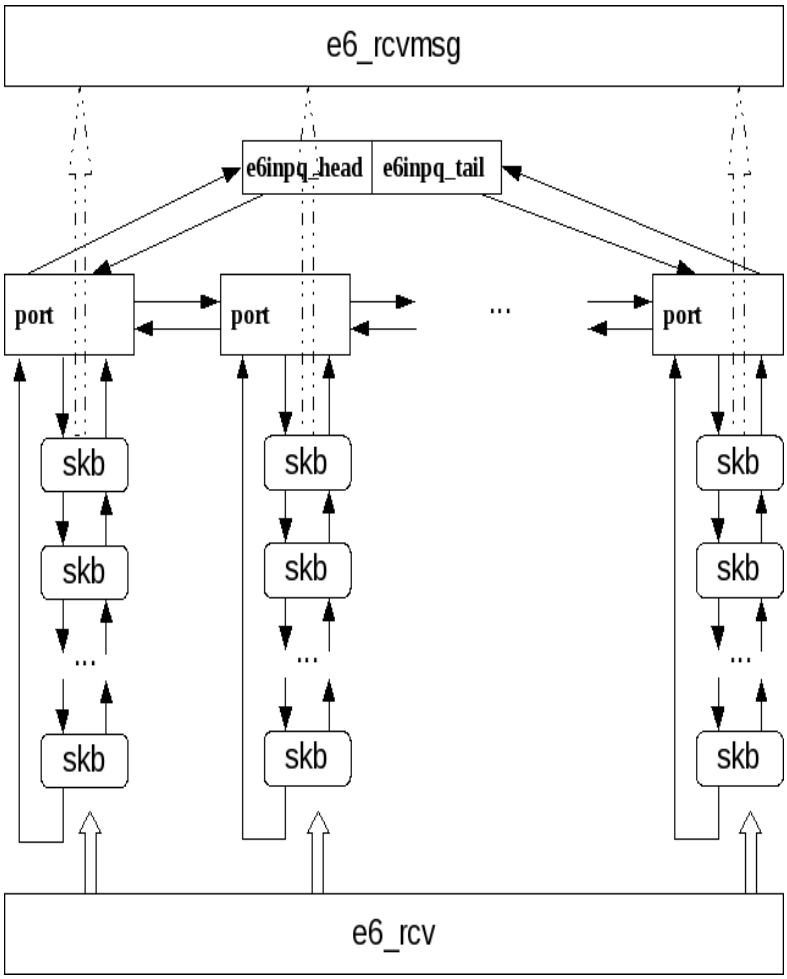


Рисунок 5.5 – Основні структури даних

```
int e6_getmsg( u16 __user *upn, struct e6msg_buf __user *msg )
```

// знаходить зареєстрований порт із номером pn=*upn

// перевіряє, чи належить порт pn поточному процесу

// перевіряє, чи порожня черга skb до порту pn

// витягає skb з голови черги

// копіює заголовок повідомлення й дані в адресний простір користувача

// звільняє skb

```
int e6_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type
*pt, struct net_device *orig_dev)
```

// витягає порт призначення dp з skb

// знаходить зареєстрований порт із номером dp; якщо відсутній, втрачає skb

// перевіряє обмеження довжини черги skb; якщо перевищено, втрачає skb

// розміщає skb у хвості черги до порту dp

// перевіряє прапор очікування й будить процес, що чекає

```
int e6_waitmsg(  u16 __user *upn )
```

// перевіряє чи зареєстровано порт pn=*upn та чи належить він поточному процесу

// перевіряє чи порожня черга skb; якщо порожня блокує поточний процес

```
int e6_regport( u16 __user *upn )
```

// перевіряє, чи зареєстрований порт pn=*upn; якщо так, повертає помилку

// створює новий запис e6portq, заповнює її й розміщає в списку зареєстрованих портів

```
int e6_unregport( u16 __user *upn )
```

// перевіряє, чи зареєстрований порт pn=*upn; якщо ні, повертає помилку

// перевіряє, чи належить порт pn поточному процесу; якщо ні, повертає помилку

// звільняє всі skb у черзі до порту pn (втрачає неприйняті пакети)

// витягає запис e6portq зі списку й звільняє її пам'ять

```
void e6udpmod_cleanup_module(void)
```

// припиняє реєстрацію типу пакетів e6: dev_remove_pack(&e6_packet_type);

// припиняє реєстрацію оброблювача переривань модуля: new_sys_e6call = NULL;

// звільняє список портів і відповідні черги skb до портів

Зауважимо, що розглянуті фрагменти коду досить спрощені для стислості викладу: опущені численні перевірки кодів повернення підпрограм і відповідні дії при виникненні помилок.

Таким чином, у дійсному підрозділі представлена перша програмна реалізація нового стека протоколів E6 у середовищі ядра версії 2.6.22.9 операційної системи Linux. Працездатність E6 мереж підтверджена успішним виконанням застосування e6talk серед інших мережних застосувань і протоколів.

Висновки по 5 розділу

1. Розроблено стратегію впровадження E6 мереж яка базується на поступовому витісненні TCP/IP мереж та містить наступні етапи: програмна реалізація стеку E6, реалізація програмних та апаратних засобів перенаправлення пакетів (маршрутизації), створення шлюзів

E6 та TCP/IP мереж.

2. Розроблено принципи функціонування шлюзів E6 та TCP/IP мереж на основі модифікованої технології трансляції адрес, побудовано схему організації шлюзів та розширену систему доменних імен для звернення до ресурсів іншої мережі з використанням суфіксу доменного імені.

3. Розроблено принципи програмної реалізації стеку протоколів E6, які передбачають реалізацію незалежну від існуючих стеків в середовищі різних операційних систем, забезпечення інтерфейсів з прикладним рівнем безпосередньо за стандартами IETF, подальше створення шлюзів з існуючими мережними технологіями.

4. Виконано експериментальну реалізацію частки специфікацій стеку E6 з прикладними інтерфейсами UDP у середовищі ядра операційної системи Linux. Відпрацьовано взаємодію з внутрішнім середовищем ядра Linux. Шляхом виконання E6 застосувань та спостереження процесів передачі E6 пакетів (кадрів) у мережі експериментально підтверджено працездатність запропонованої системи адресації E6.

ВИСНОВКИ

Розроблено нову уніфіковану систему адресації глобальних мереж, відповідний стек протоколів та алгоритмів доставки пакетів для мереж цілком побудованих на основі Ethernet з прикладними інтерфейсами сумісними з TCP/IP та гарантованою якістю обслуговування.

1 Запропоновано систему адресації E6 та відповідну корисну модель, на яку отримано патент України, що зберігає прикладне програмне забезпечення TCP/IP мереж і основні специфікації Ethernet та забезпечує гарантовану якість обслуговування завдяки використання уніфікованих адрес.

2 Розроблено новий стек протоколів E6, схему доставки пакету в E6 мережі, специфікації відповідних алгоритмів та пристроїв, які дозволяють виконати подальшу стандартизацію нової мережної технології E6.

3 Для оцінки ефективності E6 мереж порівняльне з традиційною інкапсуляцією IP повз Ethernet розроблено аналітичні та імітаційні моделі, які підтверджують істотні переваги E6 мереж стосовно підвищення корисної продуктивності мережі (майже вдвічі для телефонії) та скороченню часу доставки пакету – в 4 рази середнього та в 19 разів максимального.

4 Розроблено імітаційні моделі мереж технології PBB у формі розфарбованих сітей Петрі, які виявили певні недоліки технології PBB пов’язані із забезпечення гарантованої якості обслуговування через використання широкомовлення та додаткового відображення MAC адрес користувача і магістралі, що робить адресацію E6 унікальної з точки зору гарантованої якості обслуговування.

5 Розроблено стратегію впровадження E6 мереж, визначено основні етапи впровадження та принципи програмної реалізації стеку E6, а також основи розробки шлюзів E6 та TCP/IP мереж, що дозволяє забезпечити поступове витіснення мережами E6 інших технологій та акумуляцію їх інформаційних ресурсів.

6 Виконано експериментальну програмну реалізацію стеку E6 в ядрі операційної системи Linux, яка дозволяє спостерігати процеси передачі E6 пакетів в мережі, підтверджує працездатність E6 мереж та може бути основою для подальшої промислової реалізації стеку E6 в середовищах різних операційних систем.

РЕКОМЕНДАЦІЇ

1 Продовжити дослідження та практичну реалізацію Е6 мереж, що виведе Україну в лідери інформаційно-комунікаційних технологій.

2 Розпочати роботи із стандартизації Е6 мереж.

3 Внести напрямок стандартизації та практичної реалізації Е6 мереж до державних програм і забезпечити фінансування робіт, необхідних для повномасштабної промисловою реалізації та впровадження Е6 мереж.

ПЕРЕЛІК ПОСИЛАНЬ

1. Боллапрагада В. Структура операционной системы Cisco IOS / В. Боллапрагада, К. Мэрфи, Р. Уайт / М.: Вильямс, 2002. – 208 с.
2. Вахалия Ю. Unix изнутри / Ю. Вахалия / СПб.: Питер, 2003.– 844 с.
3. Вивек О. Структура и реализация современной технологии MPLS / О. Вивек / Вильямс, 2004.- 474 с.
4. Вирт Н. Алгоритмы и структуры данных / Н. Вирт / М.: Мир, 1989.– 360 с.
5. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский / М.: Техносфера, 2003. – 512 с.
6. Вишневский В.М. Широкополосные беспроводные сети передачи информации / В.М. Вишневский, А.И. Ляхов, С.Л. Портной, И.В. Шахнович // М.: Техносфера, 2005. – 592 с.
7. Воробиенко П.П. Всемирная сеть Ethernet? / Воробиенко П.П., Зайцев Д.А., Нечипорук О.Л. // Зв'язок, № 5, 2007. - с. 14-19.
8. Воробієнко П.П. Спосіб передачі даних в мережі із заміщенням мережного та транспортного рівнів універсальною технологією канального рівня / П.П. Воробієнко, Д.А. Зайцев, К.Д. Гуляєв / Патент України на корисну модель № 35773, u2008 03069, Заявл. 11.03.08, Опубл. 10.10.08, Бюл. № 19.
9. Гнеденко Б.В. Введение в теорию массового обслуживания / Б.В. Гнеденко, И.Н. Коваленко / М.: Наука, 1987.– 336 с.
10. Гуляев К.Д. Моделирование протоколов маршрутизации раскрашенными сетями Петри / К.Д. Гуляев // Материалы 5-й международной молодёжной научно-технической конференции «Современные проблемы радиотехники и телекоммуникаций» (РТ-2009), 20-25 апреля 2009 г., Севастополь, Украина.– с. 221.
11. Гуляев К.Д. Экспериментальная реализация стека сетевых протоколов E6 в ядре ОС Linux / К.Д. Гуляев, Д.А. Зайцев // Искусственный интеллект, № 2, 2009. – с. 105-116.
12. Дэвидсон Д. Основы передачи голосовых данных по сетям IP / Д. Дэвидсон, Д. Питерс, М. Бхатия / Вильямс, 2007.- 396 с.
13. Зайцев Д.А. Измерительные фрагменты в моделях Петри телекоммуникационных сетей / Д.А. Зайцев // Зв'язок №2(54), 2005, с. 65-71.
14. Зайцев Д.А. Исследование эффективности технологии MPLS с помощью раскрашенных сетей Петри / Д.А. Зайцев, А.Л. Сакун // Зв'язок. – 2006. – Т. 65, №5. – С. 49-55.

15. Зайцев Д.А. Оценка характеристик сетей Ethernet с помощью параметрических моделей Петри / Д.А. Зайцев, Т.Р. Шмелева // Зв'язок, № 4, 2007. - с. 62-67.
16. Зайцев Д.А. Моделирование телекоммуникационных систем в CPN Tools / Д.А. Зайцев, Т.Р. Шмелева // Одесса: ОНАС, 2009.– 72 с.
17. Керниган Б.В. Язык С / Б.В. Керниган, Д.М. Ричи / М.: Мир, 1984.– 229 с.
18. Керниган Б. UNIX – универсальная среда программирования / Б. Керниган, Р. Пайк / М.: Финансы и статистика, 1992.– 420 с.
19. Клейнрок Н. Вычислительные системы с очередями / Н. Клейнрок / М.: Мир, 1979.– 600 с.
20. Котов В.Е. Сети Петри / В.Е. Котов / М.: Наука, 1984. – 160 с.
21. Лагутин В.С. Телетрафик мультисервисных сетей связи / В.С. Лагутин, С.И. Степанов / М.: Радио и связь, 2000.– 320 с.
22. Мартин Д. АТМ. Архитектура и реализация / Д. Мартин, К. Чапмен, Д. Либен / М.: Дори, 2000.– 213 с.
23. Мурата Т. Сети Петри: Свойства, анализ, приложения / Т. Мурата // ТИИЭР. – 1989. – Т. 77, №4. – С. 41-85.
24. Олифер В.Г. Новые технологии и оборудование IP-сетей / В.Г. Олифер, Н.А. Олифер / СПб.: Питер, 2000.– 512 с.
25. Олифер В.Г. Компьютерные сети. Принципы, технологи, протоколы. / В.Г. Олифер, Н.А. Олифер / СПб.: Питер, 2006. – 958 с.
26. Питерсон Дж. Теория сетей Петри и моделирование систем / Дж. Питерсон / М. Мир, 1984. – 264 с.
27. Решения Carrier Ethernet // Компания Телеком Нетворкс: ISO 9001:2000, 2008. – 45 с.
28. Робачевский А.М. ОС UNIX / А.М. Робачевский / BHV: С-Петербург, 2001.– 528 с.
29. Слепцов А.И. Автоматизация проектирования управляющих систем гибких автоматизированных производств / А.И. Слепцов, А.А. Юрасов, под ред. Б.Н.Малиновского / К. Техніка, 1986. – 160 с.
30. Столлингс В. Беспроводные линии связи и сети / В. Столлингс / М: Вильямс, 2003.– 640 с.
31. Столлингс В. Компьютерные сети, протоколы и технологии Интернета / В. Столлингс / М: Вильямс, 2003.– 832 с.
32. Труб И.И. Объектно-ориентированное моделирование на языке С++ / И.И. Труб / С.-П.Б.: Питер, 2005. – 411 с.
33. Филимонов А. Построение мультисервисных сетей Ethernet / А. Филимонов / С.-П.Б.: Питер, 2005. – 592 с.
34. Шварц М. Сети связи: протоколы, моделирование и анализ: В 2-х ч. / М. Шварц / М: Наука, 1992. – 336 с.

35. Beaudouin-Lafon M. CPN Tools: A Tool for Editing and Simulating Coloured Petri Nets. LNCS 2031: Tools and Algorithms for the Construction and Analysis of Systems / Beaudouin-Lafon M., Mackay W.E., Jensen M. et al. // 2001, 574-580 (http://www.daimi.au.dk/CPNTools).
36. Berners-Lee T. Uniform Resource Locators (URL) / Berners-Lee T., L. Masinter, M. McCahill // Network Working Group.– 1994, RFC 1738.– 25 p.
37. Breyer R. Switched, fast, and gigabit Ethernet / Breyer R., Riley S. // MacMillan Technical Pub., 1999. – 618 p.
38. Corbet J. Linux Device Drivers / Corbet J., Rubini A., Kroah-Hartman G. / O`Reilly, 2005.– 579 p.
39. Deering S. Internet Protocol, Version 6 (IPv6) Specification / Deering S., Hinden R. // Network Working Group, 1998, RFC 2460.- 39 p.
40. Droms R. Dynamic Host Configuration Protocol / R. Droms // Bucknell University. – 1997, RFC 2131. – 45 p.
41. Duerig J. Optimizing IP Address Assignment and Static Routing at Large Scale / Duerig J., Robert Ricci, John Byers, Jay Lepreau // Proceedings of SIGCOMM 2005.
42. Egevang K. The IP Network Address Translator (NAT) / K. Egevang, P. Francis // Network Working Group. – 1994, RFC 1631. – 10 p.
43. Fang L. The Evolution of Carrier Ethernet Services – Requirements and Deployment Case Studies / Fang L., Zhang R., Taylor M. // Communications, vol. 46, no. 3, March 2008. – p. 69-76.
44. Fuller V. Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy / Fuller, V., Li, T., Yu, J., and K. Varadhan // Network Working Group. – 1993, FRC 1519. – 24 p.
45. Guliaiev K.D. Simulating E6 Protocol Networks using CPN Tools / K.D. Guliaiev, D.A. Zaitsev, D.A. Litvin, E.V. Radchenko // Proc. of Int. Conference on IT Promotion in Asia. – Tashkent (Uzbekistan), 2008. – P. 203–208.
46. Guliaiev K.D. Simulating E6 Networks Dynamic Routing / Guliaiev K.D., Zaitsev D.A. // 9th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS-9), September 17-18, 2007 - Eger, Hungary.
47. Hedric C. Routing Information Protocol / Hedrick C. // Network Working Group. – RFC 1058, 1988. – 33 p.
48. Herrin G. Linux IP Networking / Herrin G. / TR 00-04, 2000.– 100 p.
49. Hornig C. A Standard for the Transmission of IP Datagrams over Ethernet Networks / Hornig C. // Symbolics Cambridge Research Center.– 1984, RFC 894. – 2 p.
50. Hubbard K. Internet Registry IP Allocation Guidelines / K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel // Network Working Group. – 1996, RFC 2050.

51. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 2: Logical Link Control // LAN/MAN Standards Committee of the IEEE Computer Society, IEEE Std 802.2, 1998 Edition (R2003).– 239 p.
52. IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges // IEEE Computer Society, IEEE Std 801.1D, 2004. – 269 p.
53. IEEE Standard for Local and metropolitan area networks-Virtual Bridged Local Area Networks-Amendment 4: Provider Bridges // IEEE Computer Society.- IEEE Std 802.1ad, 2005.- 60 p.
54. IEEE Standard for Local and metropolitan area networks – Virtual Bridged Local Area Networks // IEEE Computer Society, IEEE Std 802.1Q, 2005. – 285 p.
55. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements. Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications // LAN/MAN Standards Committee of the IEEE Computer Society, IEEE Std 802.3, Approved 9 June 2005, IEEE-SA Standards Board IP.– 417 p.
56. IEEE Draft P802.1ah/D4.2 "Virtual Bridged Local Area Networks, Amendment 6: Provider Backbone Bridges" // Work in Progress, March 26, 2008.
57. IETF: VPLS Extensions for Provider Backbone Bridging draft-balus-l2vpn-vpls-802.1ah-03.txt // Work in Progress, July 2008.
58. Jensen K. Colored Petri Nets – Basic Concepts, Analysis Methods and Practical Use / Jensen K. // Vol. 1-3, Springer-Verlag, 1997.
59. Malkin J. RIP Version 2 Carrying Additional Information / J. Malkin // Xylogics, Inc. – 1994, RFC 1723. – 9 p.
60. Mockapetris P. Domain Names - Concepts And Facilities / P. Mockapetris // ISI. – 1987, FRC 1034. – 55 p.
61. Mockapetris P. Domain Names - Implementation And Specification / P. Mockapetris // ISI. – 1987, FRC 1035. – 55 p.
62. Plummer C. Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware / C. Plummer // MIT. – 1982, FRC 826. – 10 p.
63. Postel J. User Datagram Protocol / Postel J. // Information Sciences Institute: University of Southern California.– 1980, RFC 768.– 3 p.
64. Postel J. Internet protocol / Postel J. // Information Sciences Institute: University of Southern California.– 1981, RFC 791. – 45 p.

65. Postel J. Transmission control protocol / Postel J. // Information Sciences Institute: University of Southern California.– 1981, RFC 793.– 85 p.
66. Rekhter Y. An Architecture for IP Address Allocation with CIDR / Y. Rekhter, T.J. Watson, T. Li // Network Working Group. – 1993, FRC 1518. – 27 p.
67. Reynolds J. Assigned Numbers / Reynolds J., J. Postel // Network Working Group.– 1994, RFC 1700.– 230 p.
68. Rosen E. Multiprotocol Label Switching Architecture / E. Rosen, A. Viswanathan, R. Callon // Network Working Group, 2001, RFC 3031.– 61 p.
69. Rosen E. D. MPLS Label Stack Encoding / Rosen E, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta // Network Working Group, RFC 3032, January 2001. – 23 p.
70. Russell T. Telecommunications Protocols, 2nd Edition / Russell T. / McGraw-Hill, 2004. – 686 p.
71. Vaha-Sipila A. URLs for Telephone Calls / Vaha-Sipila A. // Network Working Group.– 2000, RFC 2806.– 21 p.
72. Zaitsev D.A. Switched LAN Simulation by Colored Petri Nets / Zaitsev D.A. // Mathematics and Computers in Simulation. – 2004. – Vol. 65, № 3. – P. 245-249.
73. Zaitsev D.A. Switched Ethernet Response Time Evaluation via Colored Petri Net Model / Zaitsev D.A., Shmeleva T.R. // Proc. of International Middle Eastern Multiconference on Simulation and Modelling, August 28-30, 2006. - Alexandria (Egypt). - 2006. - P. 68-77.

ДОДАТКИ

Додаток А. Тексти програм експериментальної реалізації стеку E6 в ядрі ОС Linux

```
*** kernel file syscall_table.S ****************************************

ENTRY(sys_call_table)
      .long sys_restart_syscall    /* 0 - old "setup()" system call, used for
restarting */
      .long sys_exit
      .long sys_fork

      . . .

      .long sys_utimensat          /* 320 */
      .long sys_signalfd
      .long sys_timerfd
      .long sys_eventfd
      .long sys_e6socketcall       /* new e6 */


*** kernel file syscalls.h **************************************************

. . .

asmlinkage long sys_socketpair(int, int, int, int __user *);
asmlinkage long sys_socketcall(int call, unsigned long __user *args);
asmlinkage long sys_e6socketcall(int call, unsigned long __user *args);
      /* new e6 */
asmlinkage long sys_listen(int, int);
asmlinkage long sys_poll(struct pollfd __user *ufds, unsigned int nfds,      long
timeout);

. . .

*** kernel file sys_ni.c ****************************************************

. . .

cond_syscall(compat_sys_ipc);
cond_syscall(compat_sys_sysctl);
cond_syscall(sys_e6socketcall);                              /* new e6 */

/* arch-specific weak syscall entries */
cond_syscall(sys_pciconfig_read);
cond_syscall(sys_pciconfig_write);

. . .

*** kernel file unistd.h ****************************************************

#ifndef _ASM_I386_UNISTD_H_
#define _ASM_I386_UNISTD_H_

/*
 * This file contains the system call numbers.
 */

#define __NR_restart_syscall      0
```

```
#define __NR_exit         1
#define __NR_fork         2

. . .

#define __NR_utimensat        320
#define __NR_signalfd         321
#define __NR_timerfd          322
#define __NR_eventfd          323
#define __NR_e6socketcall     324                     /* new e6 */

#ifdef __KERNEL__

#define NR_syscalls 325                               /* new e6 */

#define __ARCH_WANT_IPC_PARSE_VERSION
#define __ARCH_WANT_OLD_READDIR

. . .

#define __ARCH_WANT_SYS_RT_SIGACTION
#define __ARCH_WANT_SYS_RT_SIGSUSPEND

/*
 * "Conditional" syscalls
 *
 * What we want is __attribute__((weak,alias("sys_ni_syscall"))),
 * but it doesn't work on all toolchains, so we just do it by hand
 */
#ifndef cond_syscall
#define cond_syscall(x) asm(".weak\t" #x "\n\t.set\t" #x ",sys_ni_syscall")
#endif

#endif /* __KERNEL__ */
#endif /* _ASM_I386_UNISTD_H_ */

*** kernel file socket.c ************************************************

#ifdef __ARCH_WANT_SYS_SOCKETCALL

/* Argument list sizes for sys_socketcall */
#define AL(x) ((x) * sizeof(unsigned long))
static const unsigned char nargs[18]={
      AL(0),AL(3),AL(3),AL(3),AL(2),AL(3),
      AL(3),AL(3),AL(4),AL(4),AL(4),AL(6),
      AL(6),AL(2),AL(5),AL(5),AL(3),AL(3)
};

#undef AL

/*
 *    System call vectors.
 *
 *    Argument checking cleaned up. Saved 20% in size.
 *  This function doesn't need to set the kernel lock because
 *  it is set by the callees.
 */

asmlinkage long sys_socketcall(int call, unsigned long __user *args)
{

. . .
```

```
}

/* new e6 */

long (*new_sys_e6socketcall)(int call, unsigned long __user *args) = NULL;

asmlinkage long sys_e6socketcall(int call, unsigned long __user *args)
{
  int err;

  if( new_sys_e6socketcall != NULL )
    err = (*new_sys_e6socketcall)( call, args );
  else
    {
      printk(KERN_INFO"*** e6 socket.c: sys_e6socketcall echo: %d\n", call);
      err=0;
    }

  return err;
}

/* new e6 */

#endif                          /* __ARCH_WANT_SYS_SOCKETCALL */

. . .

/* ABI emulation layers need these two */
EXPORT_SYMBOL(move_addr_to_kernel);
EXPORT_SYMBOL(move_addr_to_user);

. . .

EXPORT_SYMBOL(kernel_sendpage);
EXPORT_SYMBOL(kernel_sock_ioctl);

EXPORT_SYMBOL(new_sys_e6socketcall);      /* new e6 */

*** module file e6toymod.c ***********************************************

/*
 *    e6toykit from DAZE, e6toymod.c - kernel module for e6 system calls:
e6_putmsg, e6_getmsg
 *
 *    one input/output packet/message buffers: e6msg/e6msgout
 *    the new type of e6 Ethernet frames ETH_P_E6
 *    Load: insmod e6toymod.ko [parameters]
 *    Unload:     rmmod e6toymod.ko
 *    Parameters:
 *          e6_devname - e6 device name, defaults: ¨eth0¨, example:
e6_devname=¨eth1¨
 *          e6_addr - e6 address, defaults: equals to MAC, example:
e6_addr=¨010203040506¨
 *    Requirements: needs symbol new_sys_e6socketcall exported by the kernel
 */

#include <linux/module.h>       /* Needed by all modules */
#include <linux/kernel.h>       /* Needed for KERN_INFO */
#include <asm/uaccess.h>      /* for copy_from_user */
#include <linux/skbuff.h>
#include <linux/netdevice.h>
#include <linux/rtnetlink.h>
#include <linux/moduleparam.h>
```

```
#define SYS_e6socketcall     324
#define E6_CALL_PUTMSG       1
#define E6_CALL_GETMSG       2
#define MAXE6BUFSIZE         1496
#define E6ERR_CALLNUM        -2
#define E6ERR_BUFFULL        -3
#define E6ERR_BUFEMPTY       -4
#define E6ERR_COPY           -5
#define E6ERR                -6

#define E6ADDRLEN       6
#define E6PORTLEN       2
#define E6HEADSSPACE          18
#define ETH_P_E6        0xb800

typedef unsigned char octet;

struct e6hdr {
      octet e6dst_port[E6PORTLEN];
      octet e6src_port[E6PORTLEN];
};

struct e6msg_buf {
      octet e6dst_addr[E6ADDRLEN];
      octet e6src_addr[E6ADDRLEN];
      octet e6dst_port[E6PORTLEN];
      octet e6src_port[E6PORTLEN];
      octet e6data[MAXE6BUFSIZE];
      int e6data_len;
};

static octet e6_myaddr[E6ADDRLEN];
static octet e6_zeroaddr[E6ADDRLEN]={0x00,0x00,0x00,0x00,0x00,0x00};
static char *e6_devname="eth0";
static char *e6_addr=NULL;
static struct net_device *e6_dev;
static struct e6msg_buf e6msg, e6msgout;
static int e6msg_yes = 0;

module_param(e6_devname, charp, 0000);
MODULE_PARM_DESC(e6_devname, "The name of e6 device");
module_param(e6_addr, charp, 0000);
MODULE_PARM_DESC(e6_addr, "The address of e6 device");

int e6_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type *pt,
struct net_device *orig_dev)
{
      struct ethhdr *eth;
      struct e6hdr *e6h;
      void *a;

      if (skb->pkt_type == PACKET_OTHERHOST)
      {
            kfree_skb(skb);
            return NET_RX_DROP;
      }

      /*printk(KERN_INFO"*** e6toymod: e6_rcv ***\n");*/

      a = (void *)skb->data-14;
      eth = (struct ethhdr *)a;
      memcpy( e6msg.e6dst_addr, eth->h_dest, E6ADDRLEN );
```

```
      memcpy( e6msg.e6src_addr, eth->h_source, E6ADDRLEN );

      e6h = (struct e6hdr *)skb->data;
      memcpy( e6msg.e6dst_port, e6h->e6dst_port, E6PORTLEN);
      memcpy( e6msg.e6src_port, e6h->e6src_port, E6PORTLEN);

      e6msg.e6data_len=skb->len-4;
      a = skb->data + 4;
      memcpy( e6msg.e6data, a, e6msg.e6data_len );
      e6msg_yes=1;

      kfree_skb(skb);

      return 0;
}

/* creates skb on msg and starts driver */

int e6write(struct net_device *dev, struct e6msg_buf *km)
{
  int len=km->e6data_len;
  struct sk_buff *skb;
  struct e6hdr *e6h;
  struct ethhdr *eth;
  unsigned long flags;
  int rc;

  /* alloc skb */
  skb=alloc_skb(len+E6HEADSSPACE, GFP_KERNEL);
  if (!skb) {
    printk("<1>*** e6toymod(ERR): e6write: unable alloc skb ***\n");
    return E6ERR;
  }
  skb->dev=dev;
  skb->sk=NULL;

  /* Copy data */
  skb_reserve(skb, E6HEADSSPACE);
  memcpy(skb_put(skb,len), km->e6data,len);

  /* Build the E6P2 header. */
  skb_push(skb, sizeof(struct e6hdr));
  skb_reset_transport_header(skb);
  e6h = (struct e6hdr *)skb_transport_header(skb);
  memcpy(e6h->e6dst_port, km->e6dst_port, E6PORTLEN);
  memcpy(e6h->e6src_port, km->e6src_port, E6PORTLEN);
  skb_reset_network_header(skb);

  /* Build the E6Ethernet header */
  skb_push(skb, ETH_HLEN);
  skb_reset_mac_header(skb);
  eth = (struct ethhdr *)skb_mac_header(skb);
  skb->protocol = eth->h_proto = htons(ETH_P_E6);
  if(memcmp(km->e6src_addr, e6_zeroaddr, E6ADDRLEN)==0)
    memcpy(eth->h_source, e6_myaddr, dev->addr_len);
  else
    memcpy(eth->h_source, km->e6src_addr, dev->addr_len);
  memcpy(eth->h_dest, km->e6dst_addr, dev->addr_len);
  /*printk(KERN_INFO"*** e6toymod: e6write: skb built ***\n");*/

  /* pass skb to the driver */
  local_irq_save(flags);
  netif_tx_lock(dev);
```

```
  rc=dev->hard_start_xmit(skb,dev);
  netif_tx_unlock(dev);
  local_irq_restore(flags);
  if(rc!= NETDEV_TX_OK) {
    printk("<1>*** e6toymod(ERR): e6write: hard_start_xmit failed, rc=%d ***\n",
rc);
  }

  return rc;
}

/* e6_putmsg system call */

int e6_putmsg( struct e6msg_buf __user *msg )
{
 int rc=0;

 /*printk(KERN_INFO"*** e6toymod: e6_putmsg, a0=%lu ***\n", (unsigned long)msg
);*/

 if (copy_from_user(&e6msgout, msg, sizeof(struct e6msg_buf)))
 {
      rc=E6ERR_COPY;
      return rc;
 }

 if( memcmp( e6msgout.e6dst_addr, e6_zeroaddr, E6ADDRLEN ) != 0 && memcmp(
e6msgout.e6dst_addr, e6_myaddr, E6ADDRLEN ) != 0 )
 {
   rc=e6write(e6_dev,&e6msgout);
 }
 else
 {
   memcpy( &e6msg, &e6msgout, sizeof(struct e6msg_buf) );
   e6msg_yes = 1;
 }

 return rc;
}

/* e6_getmsg system call */

int e6_getmsg( struct e6msg_buf __user *msg )
{
 int rc=0;

 /*printk(KERN_INFO"*** e6toymod: e6_getmsg, a0=%lu ***\n", (unsigned long)msg
);*/

 if( e6msg_yes == 0 )
  rc = E6ERR_BUFEMPTY;
 else
 {
   if (copy_to_user( msg, &e6msg, sizeof(struct e6msg_buf)))
      rc=E6ERR_COPY;
   else
   {
     e6msg_yes = 0;
   }
 }

 return rc;
}
```

```
extern long (*new_sys_e6socketcall)(int call, unsigned long __user *args);

long mod_sys_e6socketcall(int call, unsigned long __user *args)
{
  unsigned long a[1];
  unsigned long a0;
  unsigned char nargs = (1)*sizeof( unsigned long );
  int err;

  /*printk(KERN_INFO"*** e6toymod: mod_sys_e6socketcall %d ***\n", call );*/

  if (copy_from_user(a, args, nargs))
      return E6ERR_COPY;

  a0 = a[0];

  /*printk(KERN_INFO"*** e6toymod: args copied: a0=%lu ***\n", a0 );*/

  switch( call ){
  case E6_CALL_PUTMSG:
            err = e6_putmsg( (struct e6msg_buf __user *)a0 );
            break;
  case E6_CALL_GETMSG:
            err = e6_getmsg( (struct e6msg_buf __user *)a0 );
            break;
  default:
            err = E6ERR_CALLNUM;
            break;
  }

  return err;
}

/* system pointer to the e6 input packets handler */
static struct packet_type e6_packet_type = {
      .type = __constant_htons(ETH_P_E6),
      .func = e6_rcv,
      .gso_send_check = NULL, /*e6_gso_send_check,*/
      .gso_segment = NULL, /*e6_gso_segment,*/
};

octet e6_xdigit( char c ){octet x; if(c>='0'&&c<='9') x=c-'0'; else
if(c>='a'&&c<='f') x=c-'a'+10; else if(c>='A'&&c<='F') x=c-'A'+10; else x=0;
return x;}

static int e6toymod_init_module(void)
{
      int i, j, ok;
      int err=0;

        printk("<1>*** e6toymod: init: devname=%s ***\n", e6_devname);

      /* find & set e6_dev */
      e6_dev = dev_get_by_name(e6_devname);
      if (!e6_dev) {
         printk(KERN_ERR "*** e6toymod(ERR): init: %s doesn't exist ***\n",
e6_devname);
            return -ENODEV;
      }

      if( e6_addr != NULL )
      {
```

```
            j=0; ok=1;
            for( i=0; i<E6ADDRLEN; i++ )
            {
                  if( e6_addr[j] == '\0' ) { ok=0; break; }
                  e6_myaddr[i] = e6_xdigit(e6_addr[j++]) * 0x10;
                  if( e6_addr[j] == '\0' ) { ok=0; break; }
                  e6_myaddr[i] += e6_xdigit(e6_addr[j++]);
            }
            if( ok )
            {
                  printk("<1>*** e6toymod: init: get
e6_addr=%02x:%02x:%02x:%02x:%02x:%02x, alen=%d ***\n",
                                    e6_myaddr[0],
e6_myaddr[1],e6_myaddr[2],e6_myaddr[3],e6_myaddr[4],e6_myaddr[5], E6ADDRLEN );
                  rtnl_lock();
                  e6_dev->stop( e6_dev );

                  memcpy(e6_dev->dev_addr, e6_myaddr, e6_dev->addr_len );

                  e6_dev->open(e6_dev);
                  rtnl_unlock();

                  if (err)
                        printk(KERN_ERR "*** e6toymod(ERR): init: failed to set e6
addr, err=%d ***\n", err );
            }
      }

        if (!e6_dev->poll_controller) {
            printk(KERN_ERR "*** e6toymod: init: %s doesn't support polling
***\n", e6_devname );
      }

      if (!netif_running(e6_dev)) {
            printk(KERN_INFO "*** e6toymod: init: device %s not up ***\n",
e6_devname );
            rtnl_lock();
            err = dev_open(e6_dev);
            rtnl_unlock();

            if (err) {
                  printk(KERN_ERR "*** e6toymod(ERR): init: %s failed to open,
err=%d ***\n", e6_devname, err );
                  return err;
            }
      }

      memcpy( e6_myaddr, e6_dev->dev_addr, e6_dev->addr_len );
        printk("<1>*** e6toymod: init: myMAC=%02x:%02x:%02x:%02x:%02x:%02x,
alen=%d ***\n",
              e6_myaddr[0],
e6_myaddr[1],e6_myaddr[2],e6_myaddr[3],e6_myaddr[4],e6_myaddr[5], e6_dev->addr_len
);

      /* set net e6socketcall function */
      new_sys_e6socketcall = mod_sys_e6socketcall;

      /* register e6 packet handler */
      dev_add_pack(&e6_packet_type);

        return 0;
}
```

```
void e6toymod_cleanup_module(void)
{
        printk("<1>*** e6toymod: exit ***\n");
      dev_remove_pack(&e6_packet_type);
      new_sys_e6socketcall = NULL;
}

module_init(e6toymod_init_module);
module_exit(e6toymod_cleanup_module);

*** library header file e6toy.h ****************************************

/*
 *    e6toykit from DAZE, e6toy.h
 *
 *    The definition of e6toy application interface
 *
 */

#define MAXE6BUFSIZE          1496
#define E6ADDRLEN       6
#define E6PORTLEN       2

#define E6ERR_CALLNUM         -2
#define E6ERR_BUFFULL         -3
#define E6ERR_BUFEMPTY        -4
#define E6ERR_COPY            -5
#define E6ERR                 -6

typedef unsigned char octet;

struct e6msg_buf {
      octet e6dst_addr[E6ADDRLEN];
      octet e6src_addr[E6ADDRLEN];
      octet e6dst_port[E6PORTLEN];
      octet e6src_port[E6PORTLEN];
      octet e6data[MAXE6BUFSIZE];
      int e6data_len;
};

int e6getmsg( struct e6msg_buf * b, int * err );

int e6putmsg( struct e6msg_buf * b, int * err );

*** library file e6toy.c ***********************************************

/*
 *    e6toykit from DAZE, e6toy.c
 *
 *    The implementation of e6toy application interface
 */

#include <unistd.h>
#include <sys/syscall.h>
#include <errno.h>

#include "e6toy.h"

#define SYS_e6socketcall      324
#define E6_CALL_PUTMSG        1
#define E6_CALL_GETMSG        2

int e6getmsg( struct e6msg_buf * b, int * err )
```

```
{
 int rc;
 unsigned long a[1];

 a[0]=(unsigned long)b;

 /*printf("e6getmsg: a0=%lu \n", a[0] );*/
 rc = syscall( SYS_e6socketcall, E6_CALL_GETMSG, a );
 *err = errno;

 return rc;
}

int e6putmsg( struct e6msg_buf * b, int * err )
{
 int rc;
 unsigned long a[1];

 a[0]=(unsigned long)b;

 /*printf("e6putmsg: a0=%lu \n", a[0] );*/
 rc = syscall( SYS_e6socketcall, E6_CALL_PUTMSG, a );

 *err = errno;

 return rc;
}

*** aplication (server) file e6toyget.c ***********************************

/*
 *    e6toykit from DAZE, e6toyget.c - an application to receive e6 messages
 *
 *    Waits, gets and displays e6 messages arrived from any e6 address and any e6
ports
 *    Use Ctrl^c to stop it :-)
 *    USAGE: e6toyget
 *    (start ¨e6toyget &¨ before e6toyput for two ways communication in command
line mode)
 */

#include <stdio.h>
#include <string.h>

#include "e6toy.h"      /* application interface of e6toykit */

struct e6msg_buf buf;

main()
{
  int rc;
  int er=0;
  octet *c;
  char *s;
  unsigned int dp, sp;

  printf("e6toyget: Trying to get e6 message ... \n");

  while(1)
  {
      while ( (rc=e6getmsg( &buf, &er )) == -1 && er == 4 )
      {
            sleep(1);
```

```
      }

      if( rc != 0 )
      {
            fprintf(stderr, "*** e6getmsg failed, errno = %d\n", er);
      }
      else
      {
            dp=sp=0;
            memcpy((void *)&dp,buf.e6dst_port,E6PORTLEN);
            memcpy((void *)&sp,buf.e6src_port,E6PORTLEN);
            c=buf.e6src_addr; printf("--- From %02x:%02x:%02x:%02x:%02x:%02x(%u)
", c[0],c[1],c[2],c[3],c[4],c[5],sp);
            c=buf.e6dst_addr; printf("to %02x:%02x:%02x:%02x:%02x:%02x(%u) ",
c[0],c[1],c[2],c[3],c[4],c[5],dp);
            printf("length=%d ---\n", buf.e6data_len);
            s=buf.e6data; printf("%s\n", s);
      }

  }

}
```

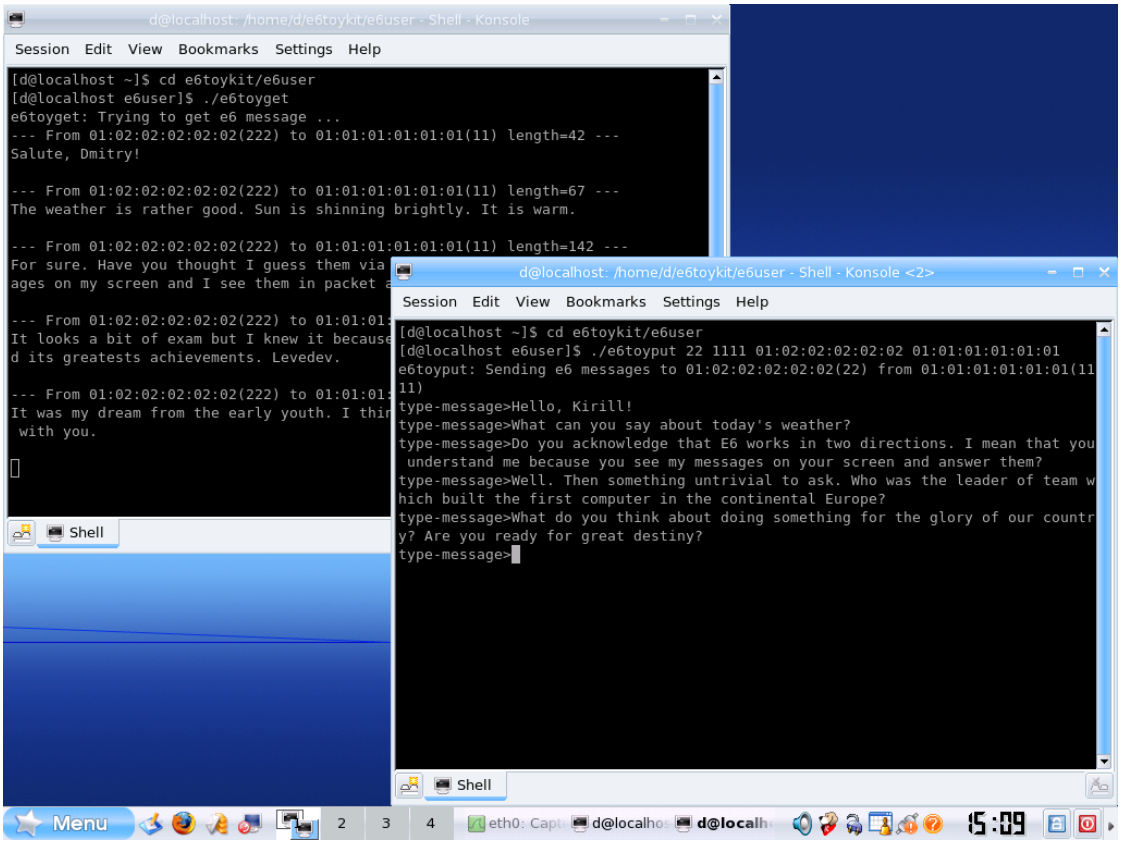
*** aplication (client) file e6toyput.c ***********************************

```
/*
 *    e6toykit from DAZE, e6toyput.c - an application to send e6 messages
 *
 *    Sends e6 messages to specified e6 address and e6 port from specified e6
address and e6 port
 *    Inputs messages from keyboard, has a default message - eternal verses
 *    Use Ctrl^c to stop it :-)
 *    USAGE: e6toyput destination_e6_port source_e6_port [destination_e6_address]
[source_e6_address]
 *    Example: ¨e6toyput 2 3 01:02:03:04:05:06¨
 */

#include <unistd.h>
#include <sys/syscall.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <netinet/ether.h>

#include "e6toy.h"      /* application interface of e6toykit */

struct e6msg_buf buf;

unsigned char data[] =
"Two households, both alike in dignity,\n"
"In fair Verona, where we lay our scene,\n"
"From ancient grudge break to new mutiny,\n"
"Where civil blood makes civil hands unclean.\n"
"From forth the fatal loins of these two foes\n"
"A pair of star-cross'd lovers take their life;\n"
"Whole misadventured piteous overthrows\n"
"Do with their death bury their parents' strife.\n"
"The fearful passage of their death-mark'd love,\n"
"And the continuance of their parents' rage,\n"
"Which, but their children's end, nought could remove,\n"
"Is now the two hours' traffic of our stage;\n"
"The which if you with patient ears attend,\n"
"What here shall miss, our toil shall strive to mend.\n";
```

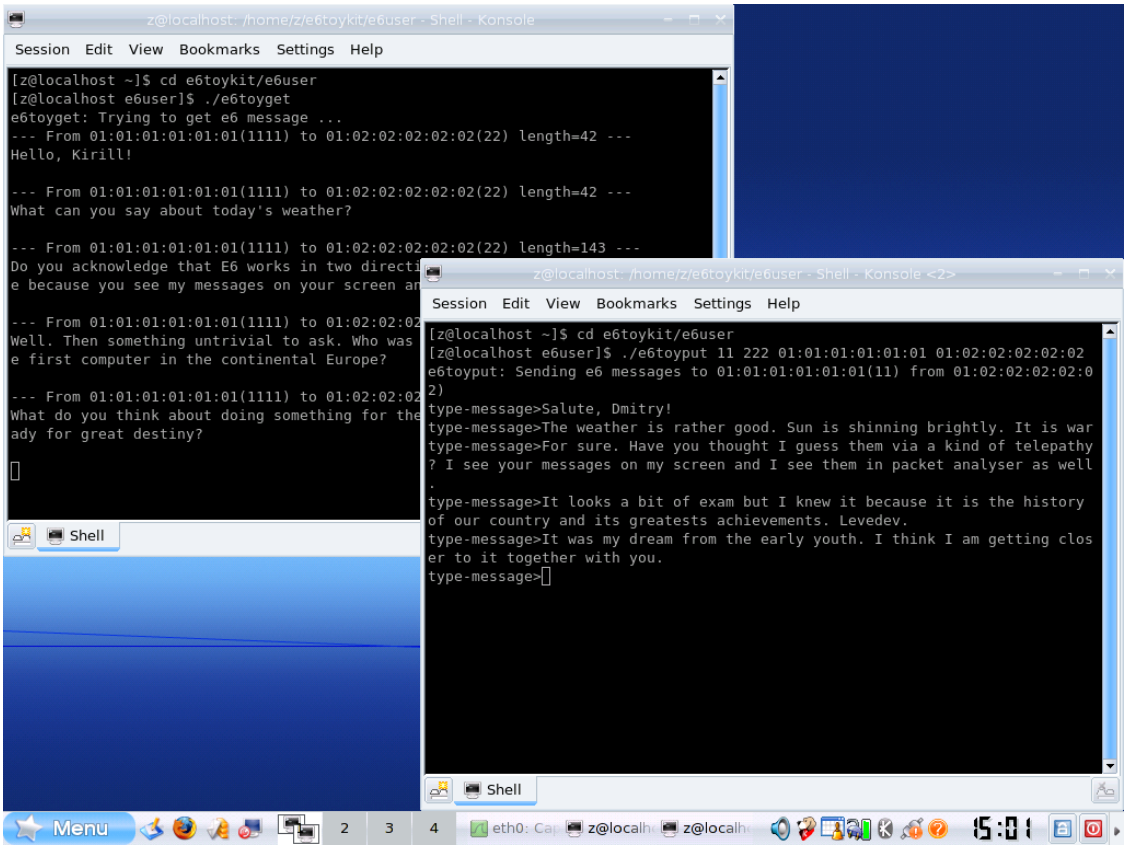
```
main( int argc, char * argv[] )
{
  int er;
  struct e6msg_buf *b = &buf;
  unsigned int dp, sp;
  struct ether_addr * ea;

  if( argc < 3 )
  {
    printf("USAGE: e6toyput destination_e6_port source_e6_port
[destination_e6_address] [source_e6_address]\n");
    return;
  }

  dp=atoi(argv[1]);
  memcpy(b->e6dst_port, (void *)&dp, E6PORTLEN );
  sp=atoi(argv[2]);
  memcpy(b->e6src_port, (void *)&sp, E6PORTLEN );
  memset(b->e6dst_addr, 0, E6ADDRLEN );
  memset(b->e6src_addr, 0, E6ADDRLEN );

  if( argc > 3 )
  {
      if( (ea=ether_aton(argv[3])) == NULL )
            printf("*** bad e6 dst address\n");
      else
            memcpy(b->e6dst_addr, ea->ether_addr_octet, E6ADDRLEN );
  }

  if( argc > 4 )
  {
      if( (ea=ether_aton(argv[4])) == NULL )
            printf("*** bad e6 src address\n");
      else
            memcpy(b->e6src_addr, ea->ether_addr_octet, E6ADDRLEN );
  }

  printf("e6toyput: Sending e6 messages to %02x:%02x:%02x:%02x:%02x:%02x(%u) from
%02x:%02x:%02x:%02x:%02x:%02x(%u)\n",
          b->e6dst_addr[0],b->e6dst_addr[1],b->e6dst_addr[2],b->e6dst_addr[3],b-
>e6dst_addr[4],b->e6dst_addr[5],dp,
          b->e6src_addr[0],b->e6src_addr[1],b->e6src_addr[2],b->e6src_addr[3],b-
>e6src_addr[4],b->e6src_addr[5],sp);

  while(1)
  {
      printf("type-message>");
      fgets ( b->e6data, MAXE6BUFSIZE , stdin );
      b->e6data_len=strlen( b->e6data )+1;
      if(b->e6data_len == 2 )
      {
            b->e6data_len=strlen( data )+1;
            memcpy(b->e6data, data, b->e6data_len);
      }
      if ( e6putmsg( b, &er ) == -1)
      {
            fprintf(stderr, "*** e6putmsg failed, errno = %d\n", er);
      }
  }

}
/*** The end ***/
```

Додаток Б. Спостереження передачі E6 пакетів (кадрів) в мережі

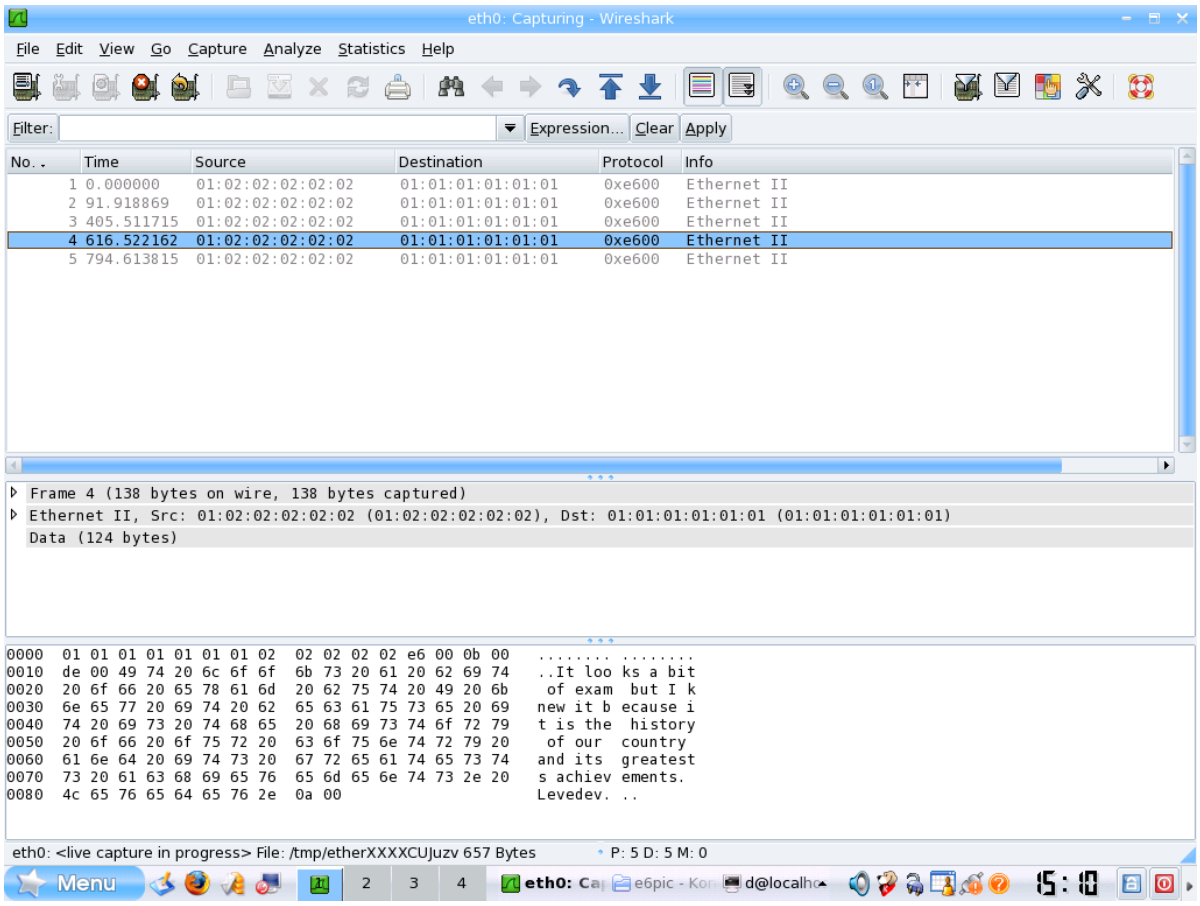
1. Образ екрану першого комп’ютера



2. Образ екрану другого комп’ютера



3. Вхідні кадри першого комп’ютеру



4. Вхідні кадри другого комп’ютеру