Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації
(повна назва)

Кафедра Радіотехнологій інформаційно-комунікаційних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти **перший (бакалаврський)**

ДОСЛІДЖЕННЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ В МЕРЕЖІ З КОМУТАЦІЄЮ ПАКЕТІВ

(тема)

Виконав:
студент **3** курсу, групи **ТРРТу-21-1**
Панов Є.І.
(прізвище, ініціали)

Спеціальність **172 Телекомунікації та радіотехніка**
(код і повна назва спеціальності)

Тип програми **освітньо-професійна**

Освітня програма **радіотехніка**
(повна назва освітньої програми)

Керівник **ст. викл. Штих І.А .**
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри _____ **Зарудний О.А.**
(підпис) (прізвище, ініціали)

2024 р.

Не містить відомостей заборонених до відкритого публікування

Керівник _____ ст. викл. Штих І.А.

Студент _____ Панов Є.І.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації

Кафедра Радіотехнологій інформаційно-комунікаційних систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма радіотехніка
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові **ПАНОВУ ЄВГЕНУ ІВАНОВИЧУ**
(прізвище, ім'я, по батькові)

1. Тема роботи **ДОСЛІДЖЕННЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ В МЕРЕЖІ З КОМУТАЦІЄЮ ПАКЕТІВ**

затверджена наказом університету від **27 травня 2024 р. № 498 Ст**

2. Термін подання студентом роботи до екзаменаційної комісії **21 червня 2024 р.**

3. Вихідні дані до роботи
Провести аналіз існуючих програмних пакетів імітаційного моделювання.
Розробити мережу для імітаційного моделювання.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ.
1. Програмні пакети для імітаційного моделювання
2. Імітаційне моделювання мережі
Висновки. Перелік джерел посилання. Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Комп'ютерна презентація – слайди у форматі Power Point

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	ст. викл. Штих І.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	06.05.2024	виконано
2	Підбір літератури за темою роботи	07.05.- 09.05.2024	виконано
3	Програмні пакети для імітаційного моделювання	10.05.- 23.05.2024	виконано
4	Імітаційне моделювання мережі	24.05.-08.06.2024	виконано
5	Оформлення презентаційного матеріалу	09.06.-10.06.2024	виконано
	підготовка до захисту у ЕК		

Дата видачі завдання **06 травня 2024 р.**

Студент ________ Є.І. Панов
(підпис)

Керівник роботи _____ ст. викл. І.А. Штих
(підпис)

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з пояснювальної записки, що містить 69 сторінок тексту, 22 рисунків, 2 таблиці, 7 джерел посилання і 3 додатки.

NETWORK SIMULATOR. ПРОГРАМНИЙ ПАКЕТ. ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ. ТОПОЛОГІЯ. OTCL.

Мета роботи – розробити мережу для імітаційного моделювання в програмному пакеті Network Simulator (NS).

Об’єкт дослідження – локальна мережа оздоровчого комплексу.

До імітаційного моделювання прибігають, коли: дорого або неможливо експериментувати на реальному об’єкті; неможливо побудувати аналітичну модель, тому що в системі є час, причинні зв'язки, наслідки, нелінійності, стохастичні (випадкові) змінні; необхідно зімітувати поведінку системи в часі.

В роботі було змодельовано мережу для оздоровчого комплесу. Дане імітаційне моделювання допомогло наглядно переконатися в роботі мережі за різних числових показників мережі. При великих навантаженнях на мережу виникаю черги та можливі втрати пакетів, але дане питання вирішується налаштуванням мережі з вищими числовими позниками.

THE ABSTRACT

The qualification work of the bachelor consists of an explanatory note containing 69 pages of text, 22 figures, 2 tables, 7 literary sources and 3 appendices.

NETWORK SIMULATOR. SOFTWARE PACKAGE. SIMULATION MODELING. TOPOLOGY. OTCL.

The purpose of the work is to develop a network for simulation modeling in the Network Simulator (NS) software package.

The object of the study is a local network of a health center.

Simulation modeling is used when: it is expensive or impossible to experiment on a real object; it is impossible to build an analytical model, because the system has time, causal relationships, consequences, nonlinearities, stochastic (random) variables; it is necessary to simulate the behavior of the system in time.

In the work, a network for a health care complex was modeled. This simulation model helped to visually verify the operation of the network under various numerical indicators of the network. Queues and possible packet loss occur when the network is heavily loaded, but this issue is resolved by setting up a network with higher numerical identifiers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ATM (Asynchronous Transfer Mode) – протокол асинхронного режиму передачі;

BGP (Border Gateway Protocol) – протокол шлюзу кордону;

CBR (Constant Bit Rate) – з постійною швидкістю передачі;

FIFO (First In, First Out) – черга, в якій «першим прийшов – першим обслугований»;

FSM (Finite-State Machine) – алгоритму роботи кінцевого автомата;

FTP (File Transfer Protocol) – протокол передачі файлів через мережу;

IP (Internet Protocol) – інтернет протокол;

ISO (International Organization for Standardization) – міжнародна організація зі стандартизації;

LDP (Label Distribution Protocol) – протокол розподілу міток;

LR-WPAN (Low-Rate Wireless Personal Area) – низькошвидкісна бездротова персональна мережа;

MDI (Multiple Document Interface) – багатодокументний інтерфейс;

MPLS (Multiprotocol Label Switching) – багатопротокольна комутація за мітками;

NS (Network Simulator) – об’єктно-орієнтований програмний продукт;

OPNET 9OPNET Modeler Suite) – засіб для проєктування і моделювання локальних і глобальних мереж, комп'ютерних систем, додатків і розподілених систем;

OSI (Open System Interconnection) – модель взаємодії відкритих систем;

OSPF (Open Shortest Path First) – протокол динамічної маршрутизації;

OSS (Open Source Code Software) – відкритий вихідний код;

OTcl (Object oriented Tool Command Language) – об’єктно-орієнтована мова скриптів (сценаріїв);

PC (Personal Computer) – персональний комп’ютер;

PPP (Point-to-Point Protocol) – двоточковий протокол канального рівня;

RSVP (Resource ReSerVation Protocol) – протокол резервування мережних ресурсів;

TCP (Transmission Control Protocol) – протокол управління передачею;

TOSSIM (TinyOS Simulator) – програмний пакет для імітаційного моделювання;

UDP (User Datagram Protocol) – протокол датаграм користувача;

VBR (Variable Bit Rate) – зі змінною швидкістю передачі;

VoIP (Voice over IP) - технологія передачі медіа-даних у реальному часі за допомогою сімейства протоколів TCP/IP;

WFQ (Weighted Fair Queuing) – зважена справедлива черга;

Wi-Fi (Wireless Fidelity) – технологія бездротової мережі;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ПП – програмний пакет.

ВСТУП

Імітаційне моделювання – це метод дослідження, при якому досліджувана система замінюється моделлю, що з достатньою точністю описує реальну систему, з нею проводяться експерименти з метою одержання інформації про цю систему [1].

До імітаційного моделювання прибігають, коли: дорого або неможливо експериментувати на реальному об’єкті; неможливо побудувати аналітичну модель, тому що в системі є час, причинні зв'язки, наслідки, нелінійності, стохастичні (випадкові) змінні; необхідно зімітувати поведінку системи в часі [1].

Імітаційне моделювання може застосовуватися у різних сферах діяльності. Особливо ефективне моделювання при вирішенні наступних завдань: проєктування та аналіз виробничих систем; оцінка різних систем озброєнь; визначення вимог до устаткування та протоколів мереж зв'язку; модернізація різних процесів у діловій сфері; аналіз фінансових і економічних систем [1].

Основним інструментом аналізу інфокомунікаційних мереж є імітаційне моделювання в комп'ютерних програмах-симуляторах без застосування реального обладнання. На етапі побудови моделі виникає питання про вибір інструментарію. При цьому важливими потребами для створення ефективної моделі є: детальна реалізація протоколів мереж; можливість написання і підключення власних модулів; можливість зміни параметрів моделювання під час проведення експериментів; платформна незалежність; розвинений графічний інтерфейс; ціна продукту [1].

1 ПРОГРАМНІ ПАКЕТИ ДЛЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

Імітаційне моделювання – це метод дослідження, при якому досліджувана система замінюється моделлю, що з достатньою точністю описує реальну систему, з нею проводяться експерименти з метою одержання інформації про цю систему [1].

Під імітаційною системою розуміють програмний або апаратно-програмний комплекс, призначений для рішення завдань із використанням методу імітаційного моделювання. При виділенні різновидів імітаційних систем виходять із того, що вони є інструментальними засобами, що забезпечують автоматизовану підтримку певних видів діяльності користувача [1].

Імітаційна система реалізує алгоритм рішення завдання і надає користувачеві сервісні можливості по керуванню обчислювальним процесом. Автоматизована підтримка інших етапів системного аналізу засобами імітаційної системи не є обов'язковою. Однак саме ступінь їхньої автоматизації визначає можливості імітаційної системи і є основою їхньої класифікації. З урахуванням етапності системного аналізу і технологічних завдань, що розв'язуються на них, виділимо можливий набір засобів імітаційної системи, що автоматизують виконання ряду функцій, реалізованих на цих етапах [1].

Створення моделі може бути підтримано наступними засобами автоматизації [1]:

– частково готовою моделлю або моделями;

– компіляторами з алгоритмічної мови високого рівня, спеціалізація якого полегшує процес складання алгоритмів імітації [1];

– спеціальною мовою високого рівня, що дозволяє виконати інформаційний або математичний опис моделі системи [1];

– конверторами моделей, що дозволяють здійснювати перетворення моделей одного виду в моделі іншого виду (інформаційної в математичну, математичної в імітаційну, інформаційної в імітаційну) [1];

– засобами контролю погодженості різних видів моделей з концептуальним поданням моделі [1].

Перевірка адекватності та технічної реалізованості може виконуватися з використанням [1]:

– програм обчислення показників адекватності;

– автоматизованої технології проведення обмеженого експерименту з імітаційною моделлю;

– програм обчислення характеристик складності моделі;

– програм обчислення ресурсних показників методу рішення завдання.

Корекція моделі може забезпечуватися [1]:

– автоматизованими технологіями редагування текстів моделей;

– програмами еквівалентних перетворень математичних і алгоритмічних моделей заданого класу.

Створення алгоритму рішення завдання може підтримуватися [1]:

– методоорієнтованими бібліотеками та пакетами програм;

– конструкторами алгоритмів рішення завдань;

– інформаційними системами підтримки прийняття рішень тощо.

Складання і уточнення схеми рішення завдання може виконуватися з використанням [1]:

– програмних засобів контролю інформаційної сумісності сімейства завдань;

– редакторів текстів обчислювальних схем тощо.

При виборі засобів імітаційного моделювання варто враховувати всі можливості, що вони надають, які можна об'єднати в наступні групи [1]:

– основні характеристики;

– сумісне програмне забезпечення;

– анімація;

– статистичні можливості;

– звіти з вихідними даними і графіками;

– послуги, що надаються замовникам і документація.

Найбільш популярними пакетами імітаційного моделювання є [2]:

– The Network Simulator (NS-2);

– OPNET Modeler Suite (OPNET);

– OMNeT++;

– Cooja;

– TOSSIM (TinyOS Simulator).

The Network Simulator (NS-2) – об’єктно-орієнтований програмний продукт, ядро якого реалізовано мовою C ++. На базі NS-2 можлива організація наочної демонстрації функціонування протоколів і мережевих механізмів [2].

OPNET Modeler Suite (OPNET) – засіб для проектування і моделювання локальних і глобальних мереж, комп'ютерних систем, додатків і розподілених систем. Включає наступні продукти: Netbiz (проектування і оптимізація обчислювальної системи), Modeler (моделювання і аналіз продуктивності мереж, комп'ютерних систем, додатків і розподілених систем), ITGuru (оцінка продуктивності інфокомунікаційних мереж і розподілених систем) [2].

OMNeT++ здатний до розширювання, модульний фреймворк, на основі компонентів та бібліотек на мові C++, який використовується для побудови моделей мереж, та являє собою симулятор дискретних подій [2].

До системи OMNeT++ закладено детальну реалізацію 24-х протоколів передачі різних рівнів моделі ISO-OSI, при чому є можливість написання і підключення власних додаткових модулів та підмодулів. У системі присутній розвинений графічний інтерфейс [2].

Більш детально розглянемо найбільш поширені пакети для імітаційного моделювання.

1.1 Програмний комплекс Network Simulator (NS-2 та NS-3)

Об'єктно-орієнтований програмний продукт Network Simulator (NS-2) має ядро, яке реалізовано на мові C ++. У якості інтерпретатора використовується

мова скриптів (сценаріїв) OTcl (Object oriented Tool Command Language). Комплекс NS-2 повністю підтримує ієрархію класів мови C++ (у термінах NS-2 зветься компільованою ієрархією) та ієрархію класів інтерпретатора OTcl (у термінах NS-2 зветься інтерпретованою ієрархією) [3].

Розробка NS почалась ще у 1989 і за останній час ця система постійно вдосконалюється. NS являє собою програму з відкритим кодом, що працює під UNIX-подібними системами. Призначення NS – освіта та дослідження в мережних технологіях. Базова структура симуляторів NS-2 та NS-3 представлена на рисунку 1.1 [3].

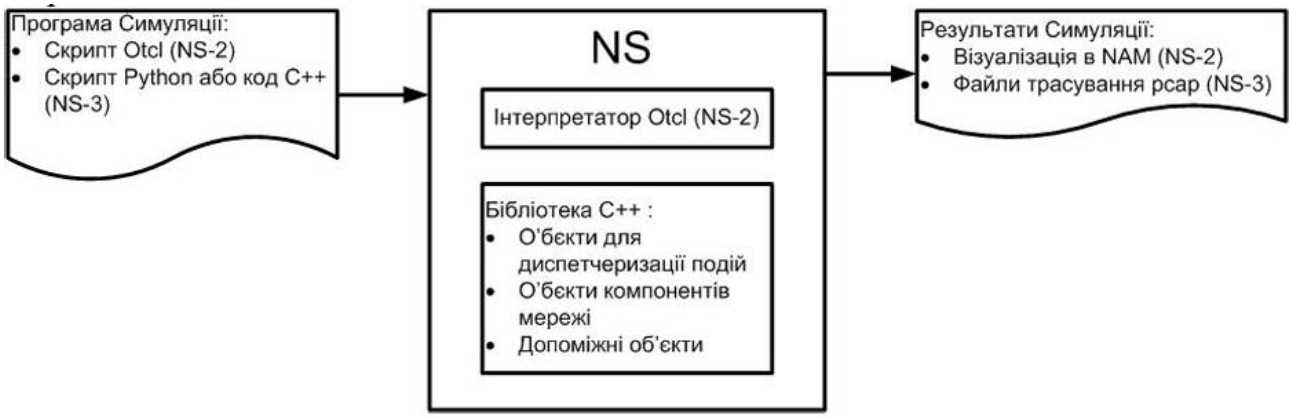

Рисунок 1.1 – Базова структура симуляторів NS-2 та NS-3 [3]

NS-3 (версія 3) є повністю новим симулятором на базі NS-2. Основна відмінність від NS-2 – відсутність OTcl (OTcl, скорочення від MIT Object Tcl), а використання програмування виключно на C++ та Python. Відмінності між NS-2 та NS-3 подано в таблиці 1.1 [3].

NS використовує дві мови, оскільки має два типи операцій для виконання. З одного боку, детальна симуляція протоколу вимагає мови програмування, яка може ефективно маніпулювати байтами, заголовками пакетів і яка водночас повинна забезпечувати можливість втілення алгоритмів, що працюють з великими масивами даних. Для цієї задачі необхідна висока швидкість виконання. Забезпечення низьких часових затрат циклу розробки

(запуск симуляції, пошук помилки в коді, виправлення помилки, ре-компіляція, повторна симуляція) є менш важливим [3].

Таблиця 1.1. – Відмінності між NS-2 та NS-3 [3]

	NS-2	NS-3
Перший реліз	1996	2008
Базова платформа	NS-1 та REAL simulator	ns-2, GTNets, YANS
Архітектура	OTcl та C++	C++ та Python
Мова скриптів	OTcl	Python
Візуалізація	NAM	ns3-viz, pyviz, nam, iNSpect
Тип симуляції	послідовна	розподілена

З іншого боку, велика кількість досліджень в галузі мережевих технологій вимагає незначних змін параметрів, змін в конфігурації мережі або швидкого перегляду можливих сценаріїв роботи. В цьому випадку згаданий вище ітераційний процес розробки є більш важливим, а швидкість виконання ролі не відіграє, оскільки конфігурування відбувається лише на початку симуляції [3].

Для цих двох різних задач NS використовує дві мови програмування. У NS-3 це C++ та Python. У NS-2 це C++ та OTcl відповідно. Тому [3]:

– OTcl – використовується для конфігурування, настройки, одноразових дій; у випадку коли існуючі об’єкти C++ повністю підходять для поставленої задачі [3];

– C++ – використовується у випадку, коли необхідна обробка пакетів або потоку; якщо з якихось причин існуючий C++ клас не підходить для поставленої задачі [3].

До прикладу, лінії зв’язку є OTcl-об’єктами, що об’єднують в собі різноманітні затримки, формування черг пакетів, моделі втрат в каналі. Якщо експеримент може бути здійснено з використанням таких об’єктів – нема

потреби у C++. Якщо ж потрібне дещо специфічне (інша організація черги, інша модель втрат) – потрібен новий об’єкт C++ [3].

На рисисунку 1.2 представлено екранну форму головного вікна додатку NS-2.

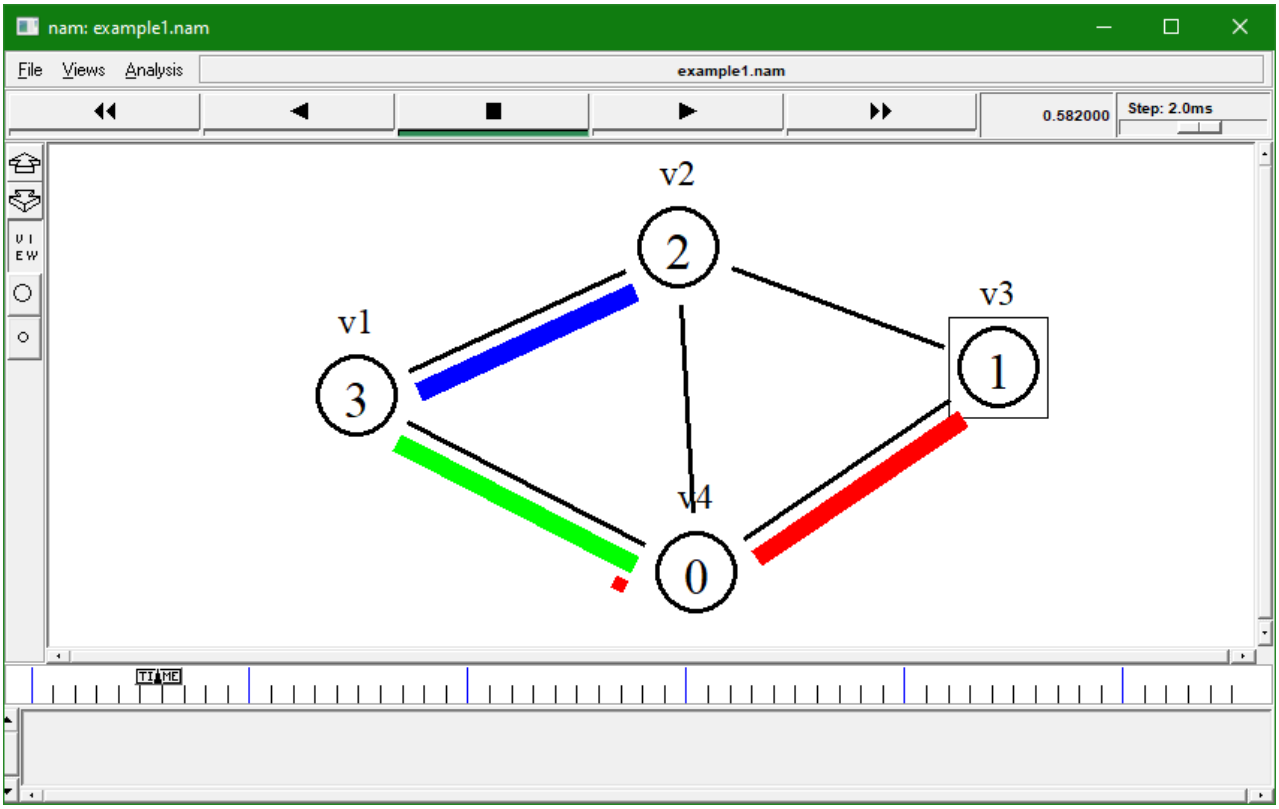

Рисунок 1.2 – Екрана форма головного вікна додатку NS-2

Використання двох мов програмування у додатку NS-2 пояснюється наступними причинами. З одного боку, для детального моделювання протоколів необхідно використовувати системну мову програмування, що забезпечує високу швидкість виконання і здатність маніпулювати досить великими обсягами даних. З іншого боку, для зручності користувача і швидкості реалізації і модифікації різних сценаріїв моделювання привабливіше використовувати мову програмування більш високого рівня абстракції. Такий підхід являє собою компроміс між зручністю використання і швидкістю. В NS-2 у якості системної мови використовується C++, що забезпечує наступне [2]:

– високу продуктивність;

– роботу з пакетам потоку на низькому рівні абстракції моделі [2];

– модифікацію ядра NS-2 з метою підтримки нових функцій і протоколів [2].

У якості мови програмування високого рівня абстракції використовується мова скриптів OTcl, яка дозволяє забезпечити ряд позитивних властивостей, які надає мова Tcl/Tk (OTcl є об'єктно-орієнтованим розширенням мови Tcl) [2]:

– простоту синтаксису;

– простоту побудови сценарію моделювання [2];

– можливість з'єднання воєдино блоків, виконаних на системних мовах програмування і просту маніпуляцію ними [2].

Об'єднання з метою спільного функціонування C++ та OTcl виконується за допомогою TclCl (Classes Tcl). TclCl являє собою інтерфейс між об'єктами мови C++ і мови OTcl, яким користуються NS-2. Однак, архітектура NS-2 не оптимальна і вимагає доопрацювань. Зокрема, поєднання у колективу модель об'єктів C++/Otcl створює деякі ускладнення: По-перше, мала поширеність і документування мови OTcl, а також складність налагодження написаних з допомогою її скриптів. По-друге, існують обмеження на спільне використання об'єктів C++. По-третє, труднощі із ефективністю використання пам'яті, симулятор повинен підтримувати обидва потоки даних - повний і обмежений [2].

На даний момент симулятор NS-2 оптимізований для обмежених потоків даних, трасування, спрощення роботи з моделями з великою кількістю елементів і зменшення обсягу реєстрованих подій, які трасуються [2],

Для отримання статистики моделювання додаток повинен гнучко конфігуруватися користувачем. Деякі можливості вже присутні в симуляторі NS-2: можливість трасування окремих пакетів або стеження за певним каналом. Більш того, необхідно дозволити користувачеві визначати тип реєстрованих даних, наприклад, чи потрібно реєструвати порти джерела і приймача при дослідженні протоколу TCP [2].

NS-2 розроблявся як програмне забезпечення з відкритим вихідним кодом (open source code software – OSS). Таке ПЗ поширюється безкоштовно – без будь- яких обмежень право використання, модифікації та розповсюдження третіми особами. З цієї ж причини безкоштовні і завжди доступні on-line всі оновлення та доповнення (нові бібліотеки, протоколи тощо) [4].

В імітаційному моделюванні мережа є сукупністю мережних об'єктів, які певним чином реагують на безліч подій. Мережеві об'єкти сповіщаються про ці події планувальником подій. Кожна подія має набір атрибутів: час наступу та мережевий об'єкт, до якого належить ця подія. Планувальник, виходячи з хронологічної таблиці, вибирає відповідні мережеві об'єкти та повідомляє про настання подій [4].

Основними компонентами моделювання NS-2 є планувальник подій, мережеві об'єкти та події (at-події та пакети) [4].

Основою моделювання в NS-2 є планувальник подій (event scheduler). Основні користувачі планувальника подій – компоненти мережі, які моделюють затримку на обробку пакетів або потрібні таймери [4].

Планувальник подій реального часу використовується для емуляції, коли пакет моделювання взаємодіє із реальною мережею [4].

Інше важливе використання планувальника подій – це планування подій моделювання, такі як, наприклад, старт програми FTP, закінчення моделювання, генерація сценарію моделювання до запуску самого моделювання [4].

Мережні об'єкти можуть бути простими (черги, лінії затримки, мультиплексори/демультиплексори, агенти, додатки) та складовими, які утворюються шляхом об'єднання простих об'єктів (вузли, дуплексні лінії, локальні мережі 802.3 Ethernet, 802.11 Wi-Fi). Вузол у NS-2 є абстракцією мережного рівня. Він може приймати пакети та класифікувати їх за адресою та портом, що є полями IP-заголовка. Лінії зв'язку є складовими об'єктами (2 протилежно спрямовані симплексні лінії, які складаються з черги, Null-агента,

лінії затримки та об'єкта). Якщо модель орієнтована на трасування подій, ланка також міститиме об'єкти трасування [4].

Агенти в NS-2 є абстракцією транспортного рівня і служать для забезпечення передачі пакетів між мережевими об'єктами. Для створення з'єднання з використанням протоколу TCP необхідна наявність двох агентів – TCP-передавачі: TCP, TCP/Tahoe, TCP/Reno, TCP/Vegas та TCP-приймач: TCPSink. Передавач має можливість відповідати по-різному на повідомлення про отримання пакетів від адресатів. Приймач приймає пакети та надсилає квитанцію про отримання. При створенні з'єднання з UDP протоколом повідомлення про отримання не передбачені, тому для передачі використовується UDP-агент, а для прийому – Null-агент, який приймає та знищує всі пакети [4].

Програми NS-2 відповідають за мережевий рівень. Додаток прикріплюється до агента та служить для створення трафіку. FTP, Telnet, CBR можна використовувати як додатки. FTP та Telnet моделюють трафік, характерний для реальних протоколів прикладного рівня. CBR є абстрактним генератором трафіку із постійним темпом видачі пакетів [4].

Типи черг задаються під час створення ланки передачі, до них относятся:

– DropTail (аналог FIFO з відкиданням пакетів, що знову надходять при переповненні черги) [4];

– FQ (рівний розподіл пропускної спроможності між потоками); SFQ (рівний розподіл пропускної спроможності між обмеженою кількістю черг); здібності між усіма класами) [4];

– CBQ (розподіл пропускної спроможності відбувається відповідно до класу пріоритету) [4].

У NS-2 пакети складаються з набору заголовків та поля даних. Формат заголовка пакета визначається під час створення об'єкта Simulator. Для зручності користувача всі пакети мають однаковий формат. Для прискорення процесу моделювання рекомендується вказувати, які протоколи будуть

використовуватися в моделі для того, щоб виключити заголовки протоколів, що не використовуються, із заголовка пакета [4].

У пакет NS-2 входить засіб анімації результатів моделювання Nam, за допомогою якого можна спостерігати роботу моделюваної мережі, її топологію, анімацію руху пакетів каналами зв'язку або їх втрати, вести візуальний моніторинг стану черг і проводити аналіз даних. Повна версія NS-2 також містить візуалізатор Xgraph, за допомогою якого можна зобразити графічно отримані результати моделювання, що знаходяться у відповідному файлі, який записуються дані в процесі симуляції [4].

У NS-2 існує і математична підтримка, що дозволяє створювати різні види трафіку, як, наприклад, трафіку пуассонівського, самоподібного і т.д.

Гнучка архітектура NS2 дає можливість її користувачам реалізовувати власні математичні функції та закони на C++ [4].

У NS2 також реалізована можливість моделювання виникнення помилок (спотворення чи втрата інформації) на канальному рівні, лише на рівні бітів чи пакетів. Є можливість реалізації своєї моделі помилок [4].

NS-2 дозволяє описати для моделі топологію, конфігурацію джерел і приймачів трафіку, параметри з'єднань (смугу пропускання, затримку, ймовірність втрати пакетів) та інші параметри. При моделюванні є можливість керувати параметрами буферів, проводити моніторинг прийнятих, відправлених та втрачених пакетів, здійснювати збір статистики та ін. За допомогою генерації вихідних tracefайлів може бути отримана інформація про динаміку трафіку, стан з'єднань та об'єктів мережі, а також роботу протоколів [4].

Для створення трафіку в системі NS-2 призначені об'єкти типу Traffic. Вони створюються методами Traffic/type, де type – Expoo, Pareto чи Trace [4].

Об'єкт Traffic/Pareto – ON/OFF генератор трафіку згідно з розподілом Парето (в NS2 є генератори та інших розподілів з «важкими» хвостами – розподіл Вейбулла, гамма-розподіл). Проста ON/OFF модель передбачає, що джерела перемикаються між двома станами: ON стан, у якому джерела

генерують трафік з постійною швидкістю, OFF-стан, у якому трафік не генерується [4].

Симулятор підтримує велику кількість протоколів, типів мереж, мережевих елементів, моделей передавання даних. Для моделювання ad-hoc мереж підтримуються протоколи маршрутизації AODV, DSDV, DSR і TORA, що вимагають додаткового доопрацювання для забезпечення можливості роботи з мобільними вузлами [3].

У симуляторі NS-2 існує модель, яка реалізує стандарт IEEE 802.15.4. Структура компонентів моделі LR-WPAN (Low-Rate Wireless Personal Area) і основні її функції представлені на рисисунку 1.3 [3].

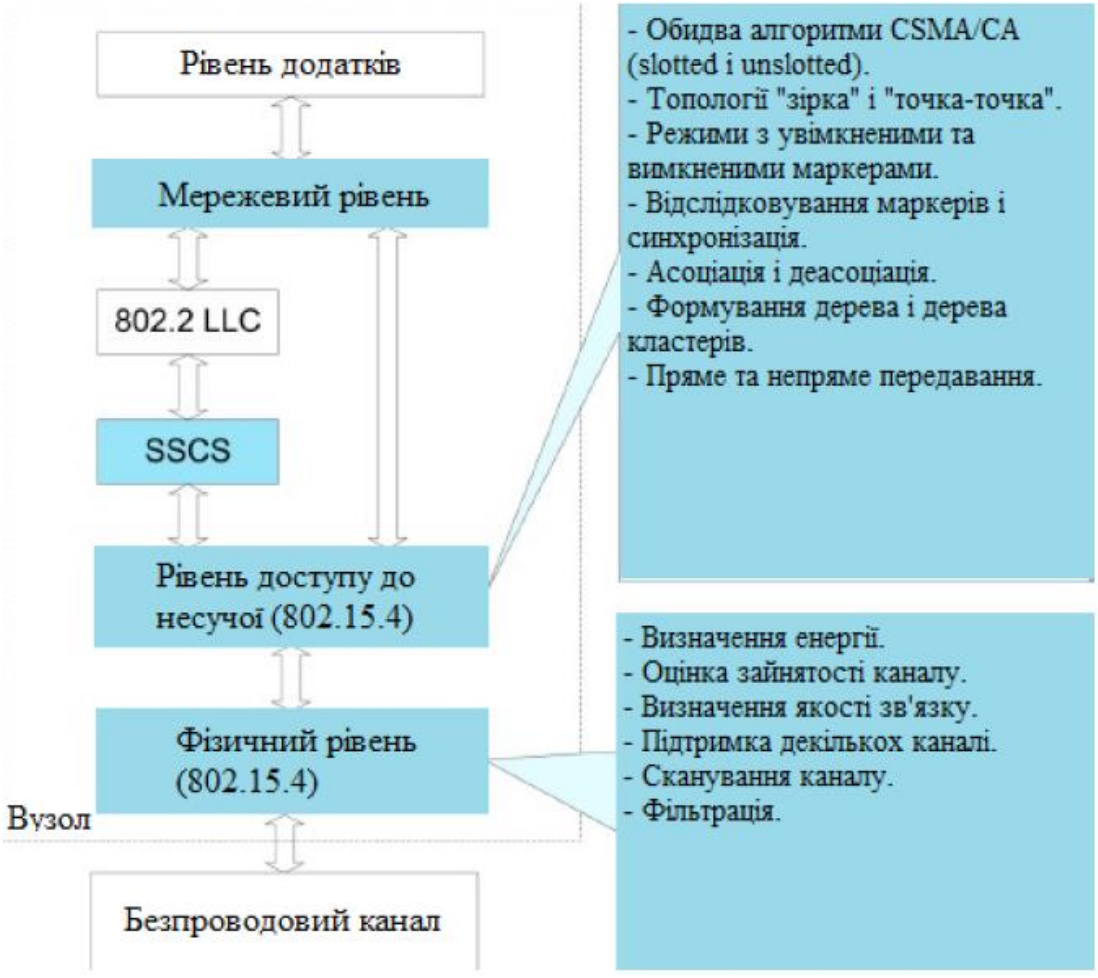

Рисунок 1.3 – Структура компонентів моделі LR-WPAN NS-2

Слід згадати, що в перших версіях моделі були реалізовані базові функції мережевого рівня ZigBee, але пізніше вони були виключені з загального

доступу, оскільки не в повній мірі відповідали даному стандарту. У зв'язку з цим на поточний момент можна використовувати тільки існуючі в NS-2 протоколи маршрутизації, які не до кінця враховують особливості безпроводових сенсорних мереж [3].

1.2 Програмний комплекс OPNET Modeler Suite (OPNET)

OPNET Modeler пропонує користувачам графічне середовище для створення, виконання та аналізу моделювання подій у інфокомунікаційних мережах. Програмне забезпечення може бути використано для великого ряду завдань, наприклад, типове створення і перевірка протоколів зв'язку, аналіз взаємодій протоколів, оптимізація та планування мереж. Також є можливість здійснити за допомогою пакетів перевірку правильності аналітичних моделей, і опис протоколів. Загальний вигляд головного вікна системи показано на рисисунку 1.4 [5].

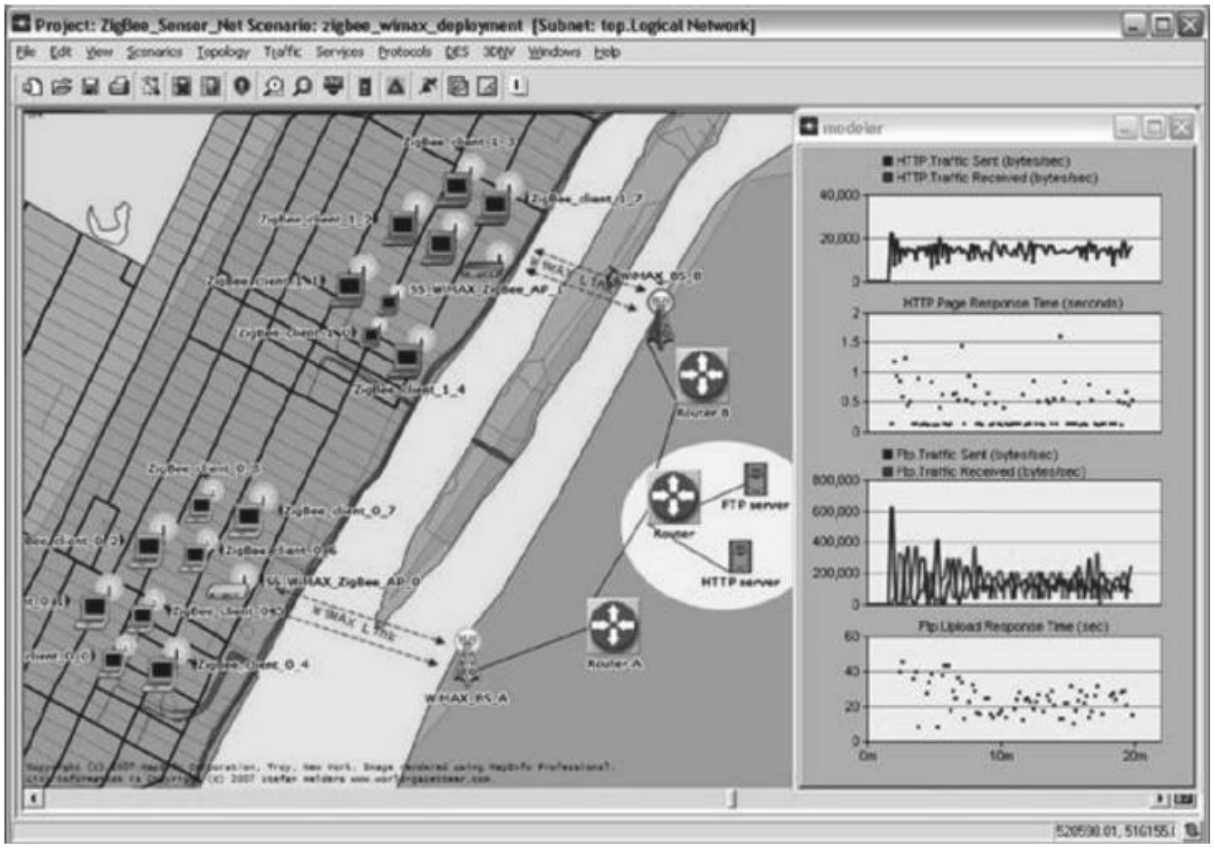

Рисунок 1.4 – Екранна форма головного вікна OPNET Modeler

Процес моделювання може бути проаналізований користувачем – додаток генерує графи і анімацію трафіка автоматично [5].

Нова особливість додатку - це автоматичне перетворення звітів у формат html. Одним з плюсів створення моделі мережі за допомогою програмного забезпечення OPNET є те, що рівень гнучкості, який забезпечується ядром моделювання, є таким же, як і для моделювання, написаних з нуля моделей, але об'єктна побудова середовища дозволяє користувачеві набагато швидше робити розробку, удосконалення та виробляти моделі для багаторазового використання. Є декілька різних середовищ у стандартному редакторі - по одному для кожного типу об'єктів. Організація цих об'єктів має ієрархічну структуру. Мережні об'єкти (моделі) пов'язують вузли мережі об'єктами комутації, при цьому, об'єкти вузлів складаються із різних типів модулів: черг пакетів, модулів процесорів, передавачів і приймачів [5].

Програмне забезпечення для моделювання радіоканалу містить моделі антени радіопередавача, антени приймача, об'єкти вузлів, яки можуть переміщуватись, включаючи супутники. Логіку поведінки процесора і модулів черг пакетів визначає модель процесу, яку користувач може створювати і змінювати в межах редактора процесу, в якому, у свою чергу, користувач може визначити модель процесу через комбінацію алгоритму роботи кінцевого автомата (finite-state machine – FSM) і операторів мови програмування C/C ++.

Виклики подій моделі процесу протягом моделювання керуються за рахунок порушення переривання, а кожне переривання, у свою чергу, відповідає події, яка повинна бути обробленою моделлю процесу [5].

Основу зв'язку між процесами представляє собою структура даних, яка називається пакетом. У додатку можуть бути задані різні формати пакета. Вони визначають, які поля можуть містити такі стандартні типи даних, як цілі числа, числа з плаваючою комою і покажчики на пакети (ця здатність дозволяють інкапсулювати моделювання пакета). Структура даних, яка викликає інформацію з контролю за інтерфейсом (interface control information - ICI), може бути розділена між двома подіями моделей процесу - це ще один механізм для

міжпроцесорного зв'язку, що зручно для команд моделювання і відповідає архітектурі багаторівневого протоколу [5].

Кожен процес, також, може динамічно породжувати дочірні процеси, що спрощує функціональний опис таких систем, як сервери. Кілька основних моделей процесу входять до базової комплектації пакета, це дозволяє моделювати популярні протоколи роботи з мережами та алгоритми, на приклад [5]:

– протокол шлюзу кордону (border gateway protocol – BGP) [5];
– протоколу контролю передачі;
– Інтернет протокол (TCP/IP);
– ретрансляції кадрів (frame relay);
– протокол Ethernet;
– протокол асинхронного режиму передачі (asynchronous transfer mode - ATM) [5];
– алгоритм WFQ (weighted fair queuing) [5].

Базові моделі корисні для швидкого розвитку складних імітаційних моделей для загальних архітектур інфокомунікаційних мереж, а також для навчання студентів, щоб дати точний функціональний опис протоколів. Існує можливість супроводу розробки коментарями і графікою (з підтримкою гіпертексту) моделей мереж, вузла або процесу. Одним з недоліків системи є велика вартість на представлений продукт [5].

1.3 Симулятор Cooja

Симулятор мережі для операційної системи (ОС) Contiki, спеціально розроблений для безпроводових сенсорних мереж, що дозволяє оцінити можливості мережі до її безпосередньої реалізації. Contiki - портативна ОС для пристроїв з низьким енергоспоживанням, таких як сенсорні вузли. Бібліотеки Contiki завантажуються і компілюються симулятором, і за допомогою певних функцій відбувається контроль і аналіз мережі. Незважаючи на те, що

симулятор розроблений для безпроводових сенсорних мереж, він також підтримує стек протоколів TCP/IP. На рисисунку 1.5 показано робоче вікно симулятора Cooja [6].

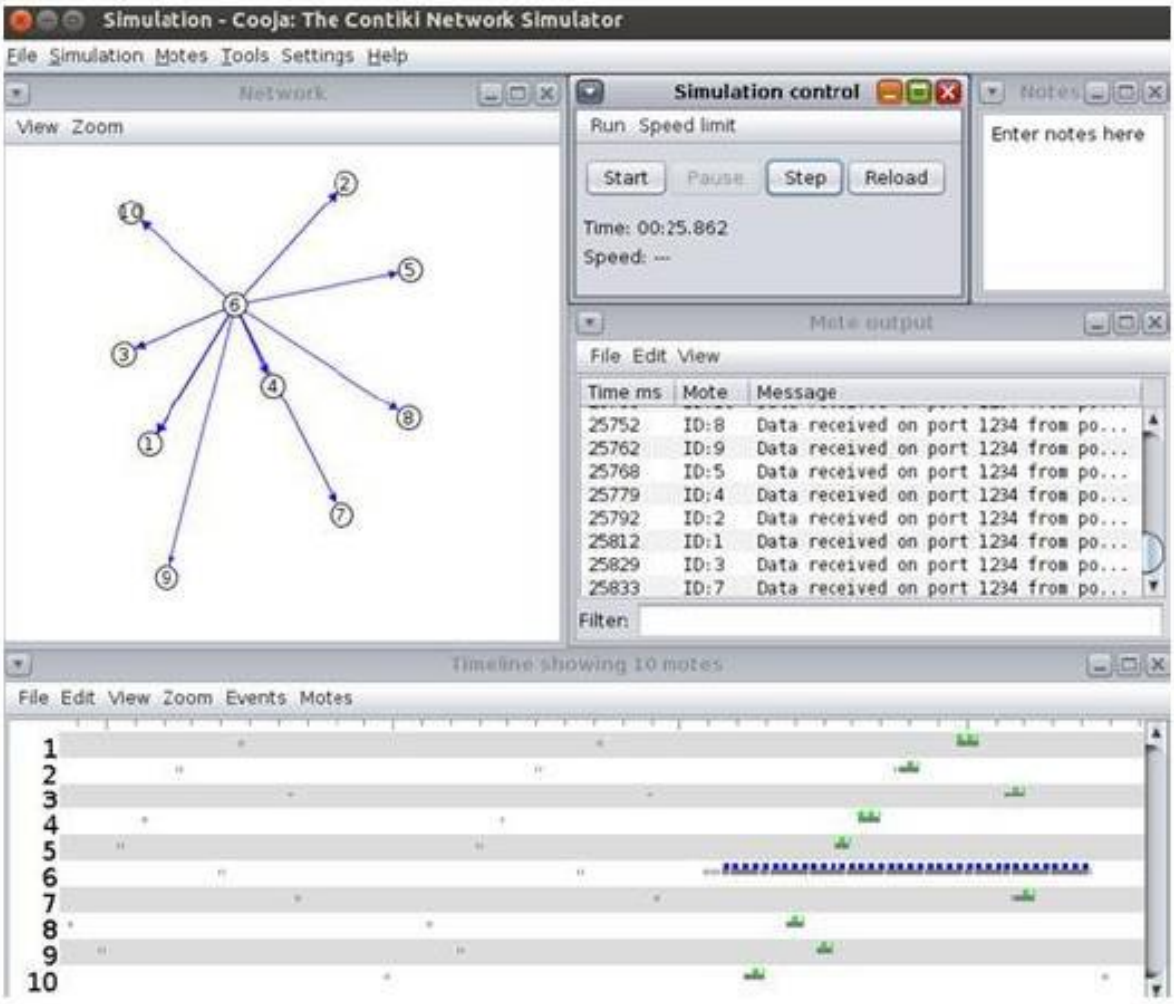

Рисунок 1.5 – Екранна форма головного вікна OPNET Modeler

Симулятор для створення моделей використовує мову Java, проте дозволяє писати програми для мережевих пристроїв на мові C [6].

Cooja є розширюваним симулятором, для цієї мети використовуються додаткові плагіни і інтерфейси. Інтерфейс описує властивості сенсорного вузла, плагіни дозволяють формувати симуляцію, наприклад, контролювати швидкість симуляції або спостерігати і контролювати трафік між сенсорними вузлами. Симулятор підтримує одночасне моделювання декількох мереж [6].

Однією з особливостей симулятора Cooja є одночасне моделювання на трьох різних рівнях – мережному рівні, рівні операційної системи і рівні інструкцій машинного коду [6].

Симулятор Cooja був розроблений для Linux і Windows/Cygwin, але пізніше з'явилася версія і для MacOs [6].

1.4 Симулятор TOSSIM (TinyOS Simulator)

TinyOS – система, спеціально розроблена для сенсорних мереж. Вона має компонентну програмну модель, описану мовою nesC. TinyOS не є операційною системою в традиційному розумінні. Це програмне середовище для вбудованих систем і набір компонентів, які дозволяють створювати імітаційні моделі за допомогою конкретних додатків, наприклад, таких як TOSSIM. Симулятор TOSSIM може моделювати мережі розмірністю до декількох тисяч вузлів, і аналізуючи їх, передбачати поведінку мережі з високою точністю. Моделюючи мережі з можливими перешкодами і помилками, симулятор створює просту, але в той же час ефективну модель взаємодій вузлів в мережі. Описуючи малопотужну модель пристроїв TinyOS, симулятор моделює поведінку сенсорного вузла з великою вірогідністю, описуючи його характеристики і проводячи велику кількість експериментів. Для зручності, TOSSIM підтримує графічний інтерфейс користувача, забезпечуючи детальну візуалізацію і відтворення дій запущеної імітаційної моделі [6].

Наведемо загальні характеристики емулятора TOSSIM [6]:

– маштабованість - симулятор підтримує модель мережі, що складається з великої кількості вузлів з різною конфігурацією. Найбільша з усіх розроблених мереж TinyOS складається приблизно з 850 вузлів, симулятор здатний підтримувати такі моделі [6];

– достовірність – симулятор описує різні взаємодії вузлів, які можуть виникнути в реальній мережі [6];

– зв'язаність - симулятор пов'язує алгоритм побудови з його графічним представленням, дозволяючи розробникам тестувати програмний код, який вимагає запуску на реальному пристрої, а також створювати візуалізації мережі [6].

Архітектура TOSSIM (рисисунку 1.6) складається з наступних елементів [6]:

– дискретний потік подій;

– набір програмних компонентів, які замінюють відповідні апаратні компоненти реальних мостів [6];

– засоби зв'язку, що надають можливість зовнішнім програмам взаємодіяти з емулятором [6].

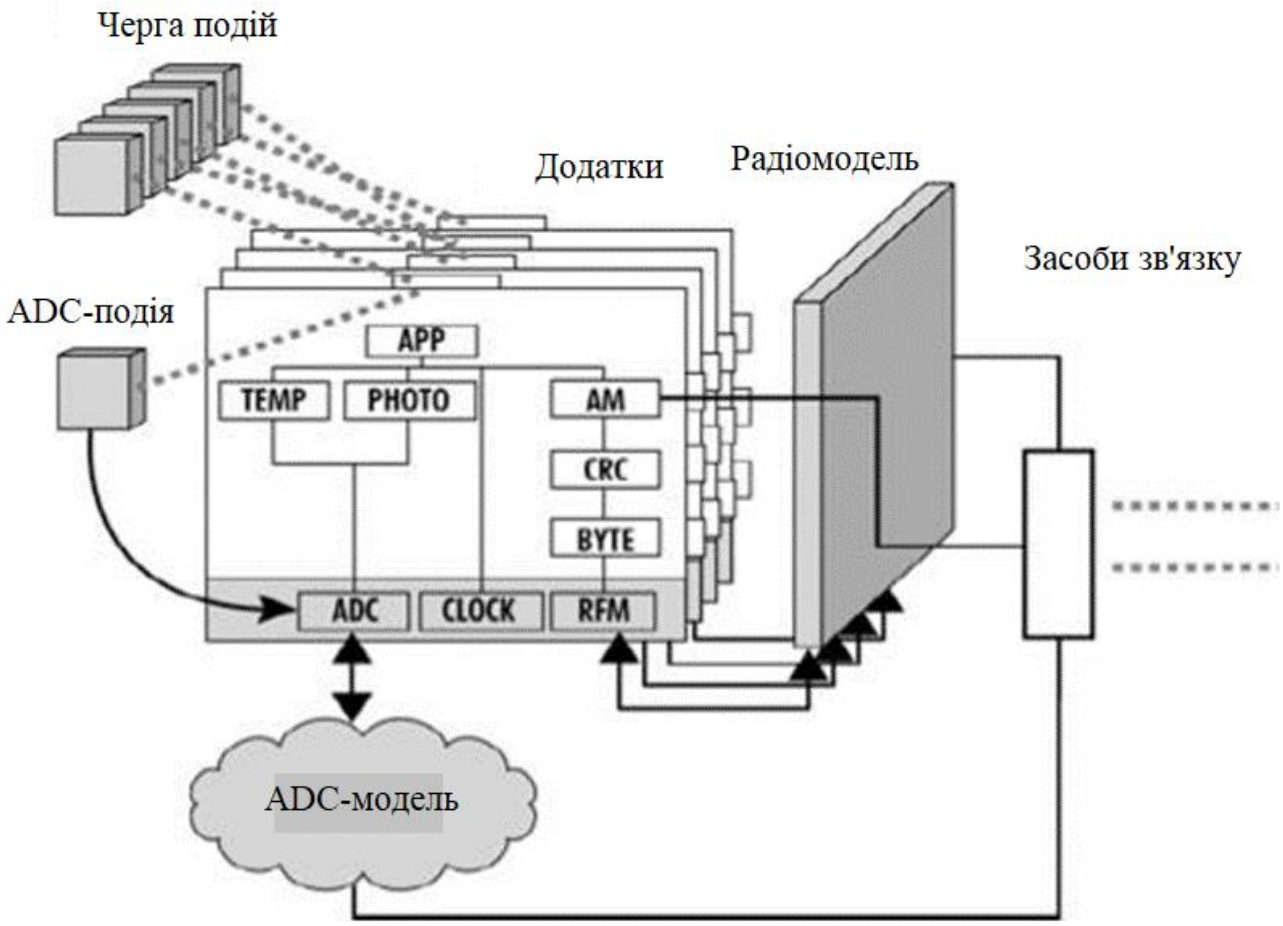

Рисунок 1.6 – Архітектура емулятора TOSSIM

1.5 Симулятор OMNeT++

Система OMNeT++ являє собою симулятор дискретних подій. Зміна стану модельованої системи відбувається в дискретні моменти часу відповідно до списка майбутніх подій (future eventlist), які відсортовано за часом. Подією може бути: початок передачі пакета, таймаут і т.п. Події відбуваються на основі виконання простих модулів (simple module). У такого модуля є функції ініціалізації, обробки повідомлення, дії і завершення роботи [7].

Система INET Framework - це комплект модулів з відкритим вихідним кодом, які дозволяють реалістично моделювати вузли мереж та протоколи дротових і бездротових мереж. Він включає моделі різних протоколів Інтернету: IP, IPv6, TCP, UDP, 802.11, Ethernet, PPP, MPLS з LDP і RSVP-TE signalling, OSPF і ряд інших. У комплект також входять різні реалістичні приклади використання цих протоколів [7].

Найвищий рівень абстракції в моделюванні IP в INET Framework предсталено мережею, яка складається з IP-вузлів. Вузол може бути маршрутизатором або хостом. IP-вузол відповідає комп'ютерному поданню стека протоколів Інтернет. Модулі, з яких він складається, організовані таким чином, щоб моделювати обробку IP-дейтаграм в операційних системах. Обов'язковим є модуль, який відповідає за мережевий рівень (який реалізує обробку IP) і модуль «мережний інтерфейс». Додатково підключаються модулі, що реалізують протоколи транспортного рівня [7].

Програма OMNeT ++ підходить для моделювання будь-якої мережі, основою якої є дискретна подія. Процес зручно відображається у вигляді об'єктів, що обмінюються повідомленнями [3].

OMNeT ++ використовує мову C ++ для імітаційних моделей. Імітаційні моделі в сукупності з мовою високого рівня NED збираються у компоненти и являються собою великі системи. Симулятор має графічні інструменти для створення моделей і оцінки результатів в режимі реального часу [3].

Моделі програми збираються з компонентів множинного використання, що називаються модулями. Модулі можна використовувати багато разів і об'єднувати за принципом блоків LEGO [3].

Модулі з'єднуються між собою за допомогою портів, і об'єднуються в складові модулі з використанням високорівневої мови програмування NED. Кількість можливих використаних модулів є обмеженою [3].

Модулі зв'язуються за допомогою передавання повідомлень, які містять довільні структури даних. Модулі можуть передавати повідомлення по певним портам і з'єднанням серверу або безпосередньо один одному. Останнє, наприклад, корисно для моделювання безпроводових мереж [3].

В основному вікні візуалізації (рисисунку 1.7) показано комп'ютерну мережу для якої проводять моделювання [3].

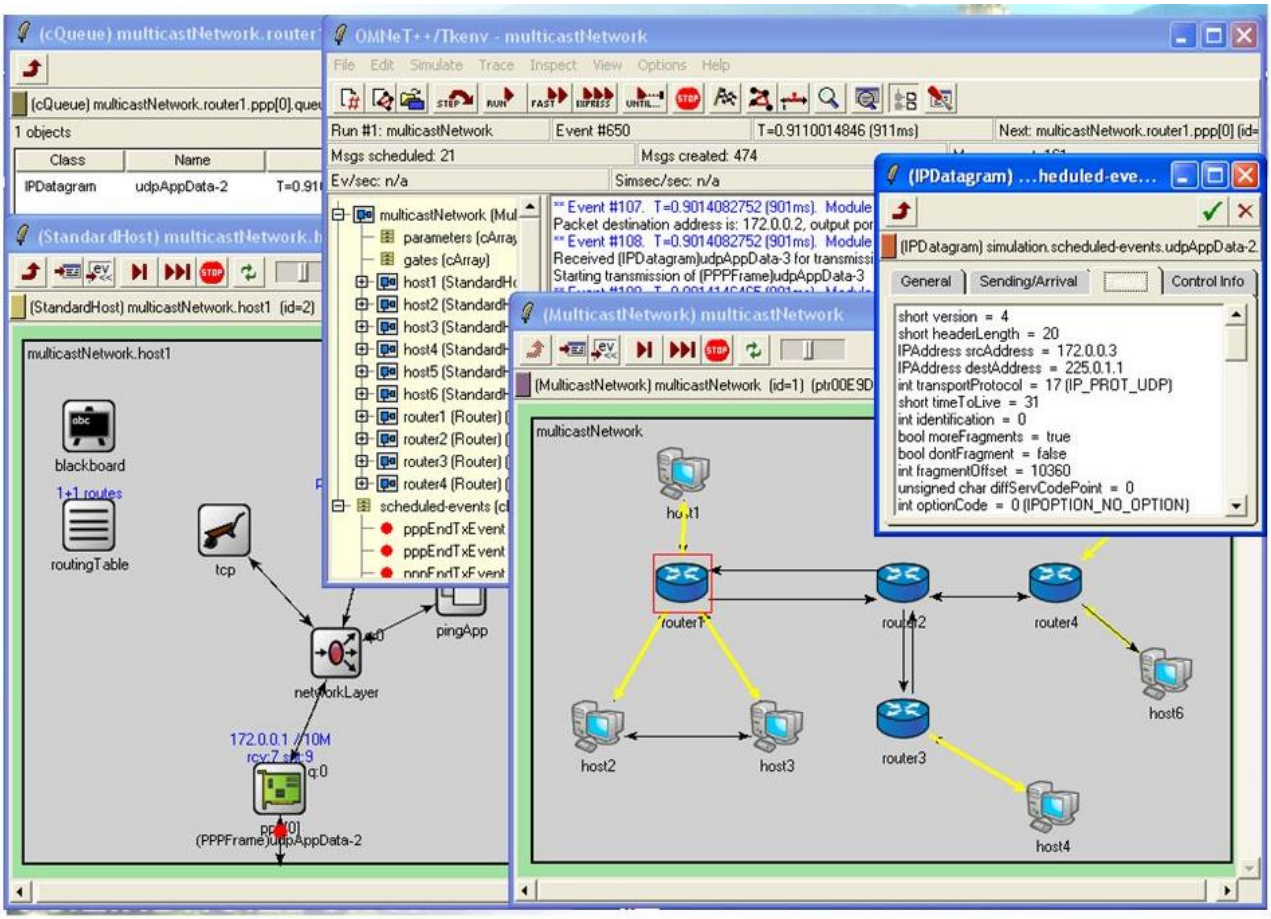

Рисунок 1.7 – Екранна форма головного вікна OMNeT++

Вона являє собою набір клієнтських станцій, сервер і маршрутизатор, з'єднаних каналами зв'язку. Кожен вузол може нести різну функціональність в залежності від параметрів або набору внутрішніх модулів. Внутрішні модулі відповідають за роботу протоколів і додатків на різних рівнях моделі OSI [3].

Вузли мережі з'єднуються між собою каналами зв'язку, параметри котрих можна змінювати. На основному вікні візуалізації відображається внутрішня структура сервера і маршрутизатора (протоколи мережного, транспортного рівнів моделі OSI, таблиця маршрутизації і вказано запис логу всіх процесів, які відбуваються. Всі вищезгадані властивості доступні в будь-який час протікання процесу. Тим самим, користувач може відстежити пересування пакета, який його цікавить, по всій мережі [3].

Якщо реалізація протоколів в NS-2 доступна в публічному користуванні, то застосування того ж протоколу в OMNeT ++ має складності, так як архітектура цих симуляторів різна [3].

Процес моделювання може працювати в різних призначених для користувача інтерфейсах. Графічно анімований призначений для користувача інтерфейс зручний для демонстрації та налагодження мережі, а інтерфейс командного рядка зручний для внесення змін [3].

Компоненти OMNeT ++ [3]:

– коренева бібліотека моделювання;

– OMNeT ++ IDE на базі платформи Eclipse;

– графічний інтерфейс виконуваного моделювання, посилання на виконуваний файл (Tkenv) [3];

– призначений для користувача інтерфейс командного рядка для виконання моделювання (Cmdenv) [3];

– документація, приклади.

В таблиці 1.2 наведені загальні порівняльні характеристики можливостей симуляторів NS-2 і OMNeT ++.

Таблиця 1.2 – Порівняльні характеристики можливостей NS-2 і OMNeT++ [3]

Параметри	NS-2	OMNeT++
Гнучкість	NS-2 було розроблено в якості TCP/IP, відповідно його використовують для імітації мереж з пакетним передавання даних. NS-2 має жорсткі уявлення про вузли мережі, мережеві протоколи, представлення пакетів, мережеві адреси, що має свої переваги, та не дає вносити будь-які зміни.	OMNeT++ має гнучку структуру моделювання. Можна змоделювати будь-яку мережу, компоненти якої взаємодіють шляхом передавання повідомлень.
Синхронізація	Дискретні події	Дискретні події
Платформа системи моделювання	Linux, FreeBSD, Solaris, Windows (Cygwin)	Linux, Unix, Windows (Cygwin)
Підтримка графічного інтерфейсу	Моніторинг потоку симуляції	Моніторинг потоку симуляції, розробка і визначення топології на C++, результат аналізу і симуляції
Документація	Документація NS-2 фрагментована, мало навчальної літератури.	OMNeT++ має доступну документацію, велику кількість навчальної літератури, відеоуроки.
Масштабованість для мереж великих розмірів	В NS-2 великі мережі складно масштабовані.	OMNeT++ Підтримує моделювання великих мереж. Обмежується виключно можливостями комп’ютера, на якому виконується симуляція.

1.6 Інші симулятори

Основним призначенням симулятора javaNetSim є імітація роботи всіх рівнів стека протоколів TCP/IP. Для цього в ньому імітується робота протоколів кожного з рівнів, чим досягається повна імітація роботи мережі. Таким чином симулятор javaNetSim є зручним для виконання лабораторних робіт студентів [7].

Симулятор javaNetSim є об'єктно-орієнтованим, він написаний мовою Java, є кросплатформним, тобто javaNetSim працює під будь-якомою операційною системою, де є віртуальна Java-машина. На рисисунку 1.8 представлено вікно програми [7].

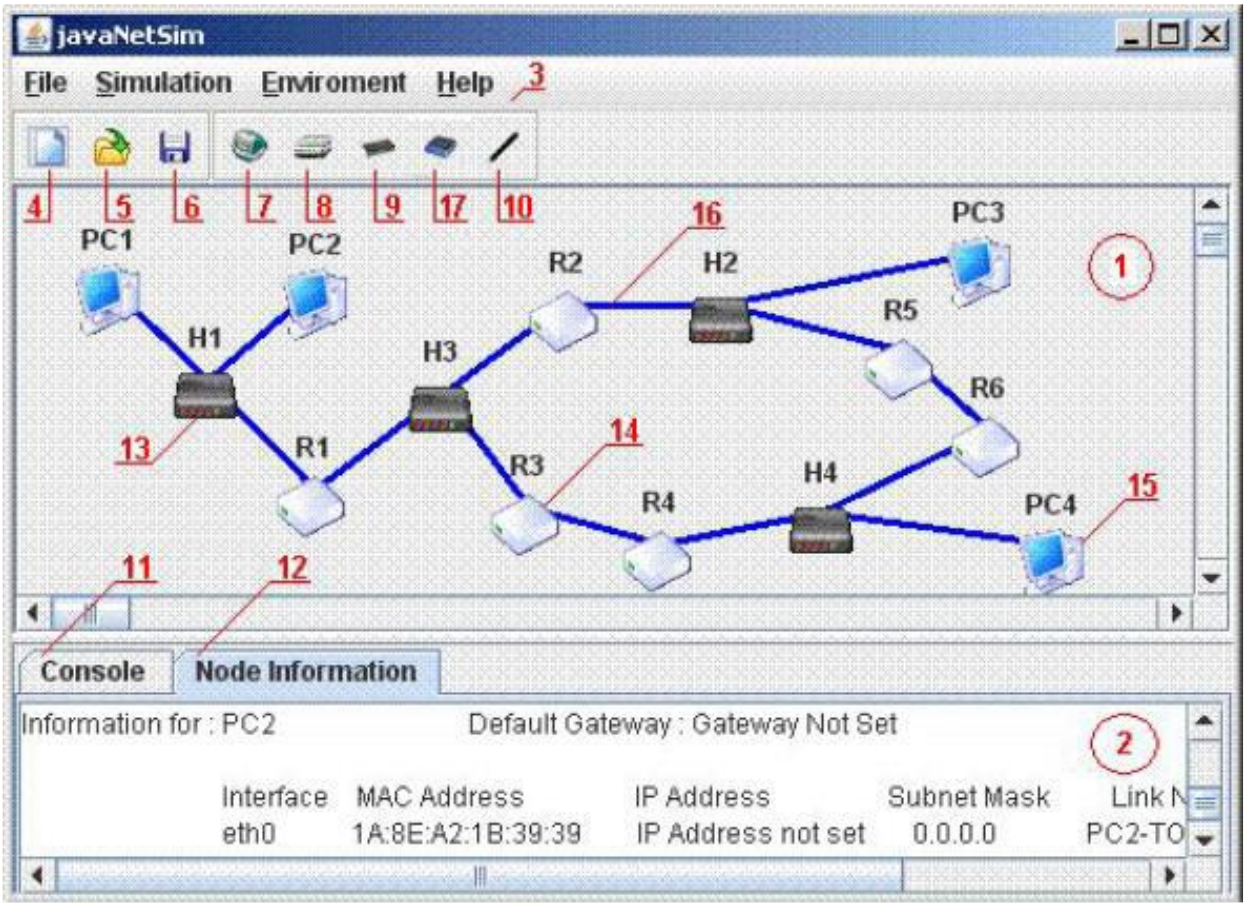

Рисунок 1.8 – Екранна форма головного вікна javaNetSim

Архітектура симулятора javaNetSim має наступний вигляд. В його основі лежить клас Simulation (Імітація), який містить об'єкти класів Link (Лінія

зв’язку) і Node (Вузол мережі). Цей клас призначений для об'єднання пристроїв і ліній зв'язку в єдину мережу. Клас Link містить канали зв’язку для об'єктів класу Node, і призначений для з'єднання двох вузлів між собою [7].

Клас Node містить зв’язки з об'єктами класу Link і є найбільш загальною моделлю мережевого пристрою [7].

Всі реальні мережеві пристрої є похідними від об'єкта класу Node і відповідають моделі стека протоколів TCP/IP [7]:

– Hub (Концентратор) – DataLink Layer Device (пристрій фізичного рівня) - має п'ять портів, тобто до нього можливо підключити до п'яти ліній зв'язку [7];

– Router (Маршрутизатор) – Network Layer Device (пристрій мережевого рівня) - має два порти, а також стек протоколів TCP / IP (ProtocolStack) [7];

– PC (ПК) – Applications Layer Device (пристрій рівня додатків) - має один порт, стек протоколів TCP/IP, а також можливість виконувати клієнтську або серверну частину будь-якої програми [7].

Для взаємодії з користувачем та візуалізації кожному мережевому пристрою потрібний графічний відповідник. Його забезпечують наступні класи [7]:

– GuiHub (графічний інтерфейс концентратора);

– GuiRouter (графічний інтерфейс маршрутизатора).

Як самі мережеві пристрої, так і графічний користувальницький інтерфейс мережевих пристроїв повинен бути єдиним. Цим об'єднанням займається клас SandBox (Робоча область) [7].

Система імітаційного моделювання мереж COMNETIII дозволяє точно прогнозувати продуктивність локальних, глобальних і корпоративних мереж. Система працює в середовищі Windows і Unix [2].

COMNETIII пропонує використовувати простий і інтуїтивно зрозумілий спосіб конструювання моделі мережі, заснований на застосуванні готових базових блоків, відповідних добре знайомим мережевих пристроїв, таким як комп'ютери, маршрутизатори, комутатори, мультиплексори і канали зв'язку [2].

Користувач застосовує техніку drag-and-drop для графічного зображення модельованої мережі з бібліотечних елементів. Потім система COMNETIII виконує детальне моделювання отриманої мережі, відображаючи результати динамічно у вигляді наочної мультиплікації результуючого трафіку [2].

Після закінчення моделювання користувач отримує в своє розпорядження наступні характеристики продуктивності мережі [2]:

– прогнозовані затримки між кінцевими і проміжними вузлами мережі, пропускні спроможності каналів, коефіцієнти використання сегментів, буферів і процесорів [2];

– піки і спади трафіку як функцію часу, а не як усереднені значення;

– джерела затримок і вузьких місць мережі.

У COMNETIII моделюється не тільки взаємодія комп'ютерів по мережі, але і процес поділу процесора кожного комп'ютера між його додатками. Робота програми моделюється за допомогою команд декількох типів, в тому числі команд обробки даних, відправки та читання повідомлень, читання і запису даних в файл, встановлення сесій і припинення програми до отримання повідомлень.

Канали зв'язку моделюються шляхом завдання їх типу, а також двох параметрів - пропускної здатності і вноситься затримки поширення. Одиницею переданих по каналу даних є кадр. Пакети при передачі по каналах сегментируются на кадри. Кожен канал характеризується: мінімальним і максимальним розміром кадру, накладними витратами на кадр і інтенсивністю помилок в кадрах [2].

COMNETIII дозволяє при моделюванні задавати форму звіту про результати для кожного окремого елемента моделі. Звіт генерується кожен раз при запуску певної моделі. Звіт представлений в стандартній текстовій формі, і його легко можна роздрукувати на будь-якому принтері. Можна задати генерацію кількох звітів різного типу для кожного елемента мережі [2].

Існують інші способи отримання статистичних результатів прогону моделі, крім звітів. У COMNETIII є кнопки Statistics, за допомогою яких можна

включити збір статистики для кожного типу елемента моделі - вузлів, каналів, джерел графіка, маршрутизаторів, комутаторів і т. Д. Монітор статистики кожного елемента можна встановити для збору тільки базових статистичних параметрів (мінімум, максимум , середнє значення і дисперсія) або ж збору даних в тимчасовому масштабі для побудови графіків. Якщо результати спостережень збережені у файлі для подальшої побудови графіків і аналізу, то можливо також побудова гістограм і процентних показників [2].

Будівник звітів, монітор статистики, апарат візуалізації є конкурентними і відмітними рисами цієї розробки [2].

IxChariot - це інструмент на базі програмного забезпечення для оцінки мереж, використовуваний для виміру ключових функціональних характеристик, таких як пропускна спроможність, година затримки, втрата пакетів, jitter, MOS для VoIP і MDI для відео в реальних умовах, і використовується щодня провідними компаніями і випробувальними лабораторіями для атестації і сертифікації новітніх мережевих пристроїв [2].

Виміри робочих характеристик проводяться шляхом передачі реальних потоків даних між прибудовами, підключеними до мережі. IxChariot емулює (рисисунку 1.9) різні типи розподілених застосувань, збирає і аналізує отримані результати. Кінцеві точки IxChariot генерують трафік, використовуючи ті ж методи, що і будь-яке мережеве застосування, дозволяючи виміряти кожен елемент в тракті передачі даних [2].

Статистичної інформації тільки по мережевому рівню недостатньо для того, щоб передбачити робочі характеристики прикладного рівня в корпоративних мережах і широкосмугових транспортних мережах. Наприклад, розуміння впливу втрати пакетів має значення тільки в контексті конкретних мережевих застосувань. IxChariot® можна використати для досягнення максимальної продуктивності мережі і пристроїв [2].

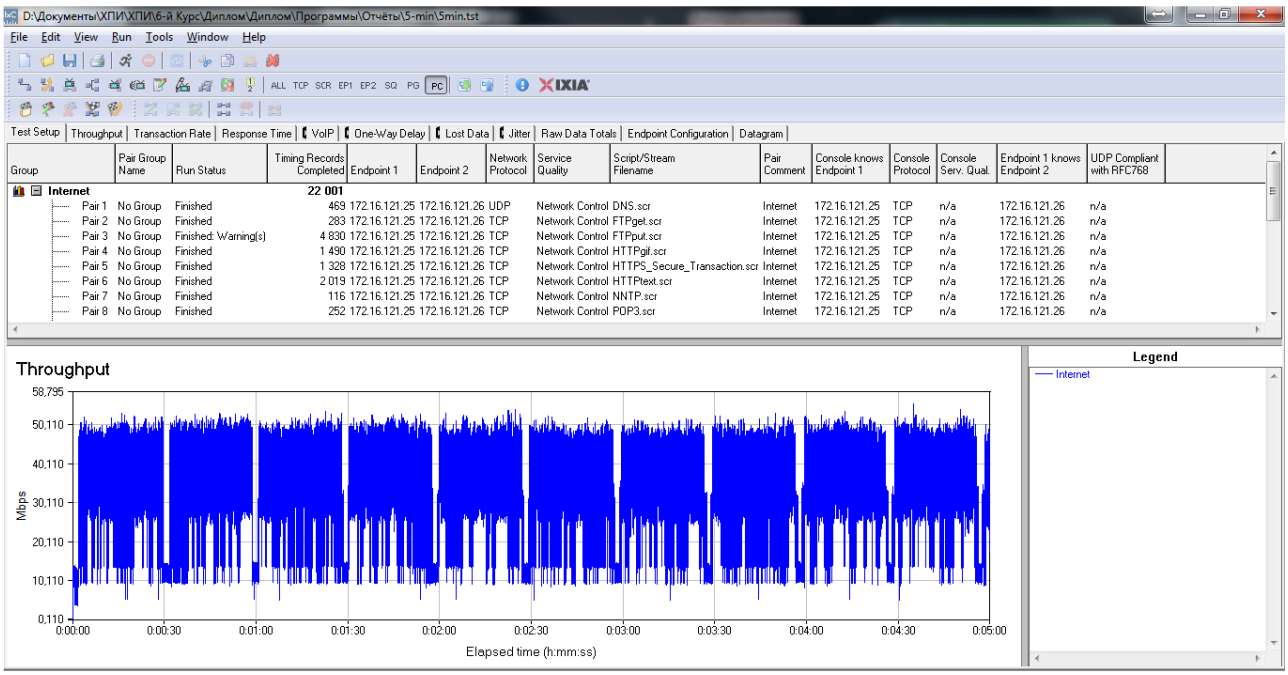

Рисунок 1.9 – Екранна форма головного вікна IxChariot

Wireshark (раніше — Ethereal) — програма-аналізатор трафіку для комп'ютерних мереж Ethernet і деяких інших. Має графічний інтерфейс призначений для користувача. Функціональність, яку надає Wireshark, дуже схожа з можливостями програми tcpdump, проте Wireshark має графічний призначений для користувача інтерфейс і значно більше можливостей по сортуванню і фільтрації інформації. Програма дозволяє користувачеві переглядати увесь трафік, що проходить по мережі, в режимі реального часу, переводячи мережеву карту в нерозбірливий режим.(англ. promiscuous mode) [2].

Програма поширюється під вільною ліцензією GNU GPL і використовує для формування графічного інтерфейсу кроссплатформенную бібліотеку GTK+ (планується перехід на Qt) Існують версії для більшості типів UNIX, у тому числі Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, а також для Windows [2].

Wireshark — це застосування, яке «знає» структуру найрізноманітніших мережевих протоколів, і тому дозволяє розібрати мережевий пакет, відображаючи значення кожного поля протоколу будь-якого рівня. Оскільки

для захоплення пакетів використовується WinPcap, існує можливість захоплення даних тільки з тих мереж, які підтримуються цією бібліотекою. Проте, Wireshark уміє працювати з множиною форматів вхідних даних, відповідно, можна відкривати файли даних, захоплених іншими програмами, що розширює можливості захоплення [2].

AutoSignal 1.7 є першим і єдина програма, яка повністю автоматизує процес аналізу сигналів. Це надає дослідникам право швидко знайти компоненти складних сигналів, які зазвичай вимагають широкого програмування і математичних функцій. Це повною мірою її графічного інтуїтивно зрозумілий інтерфейс користувача, щоб спростити усі аспекти роботи, з імпорту даних для виводу результатів [2].

Авторегресійних лінійні моделі пропонують надійні моделі, які можуть швидко обробляти менше наборів даних, які ШПФ (швидке перетворення Фур'є) не може точно аналізувати. З AutoSignal, ви можете також відновити компоненти сигналу на підставі потужності - компонент може бути синусоїдальним, меандр, пилкоподібний або ангармонічний малюнок. Вона дозволяє побачити повну картину частотною простору з використанням бібліотеки шести методів Фур'є спектру з повною гнучкістю [2].

Це дає вам вибір з трьох регульованих матерій сплесків : Морле, Павла і гаус похідних - в реальних і складних формах для оптимізації результатів локалізації [2].

Учені і інженери можуть виконувати комплексний аналіз сигналу без програмування, вибравши пункти меню, які визначають, як комп'ютер аналізуватиме і представлятиме дані. Вибір етапу обробки або алгоритму викликає до програмного забезпечення для подальших меню або вікон, так що користувачі можуть адаптувати продуктивність на їх потребі. Після того, як користувач робить вибір обробки, програмне забезпечення відразу представляє результати обробки в 2 - D або 3 - D виді [2].

2 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ МЕРЕЖІ

2.1 Побудова мережі

Кожна комп’ютерна мережа має свою логічну та фізичну топології. Фізична топологія вказує на те, яким чином робочі станції з’єднуються між собою використовуючи один із видів зв’язку.

Логічна топологія передбачає набір певних правил, які існують у мережі. Ці правила регламентують взаємодію між робочими станціями. Недотримання їх призведе до порушення роботи мережі в цілому. Розглянемо більш детально типи логічних та фізичних топології з вказання переваг та недоліків кожної з них.

Головне завдання, яке ставиться перед мережею, яка розробляється – «один для всіх» і «всі для одного». Це значить, що в мережі є головний ПК, який надсилатиме інформацію для всіх решти, а також вся інформація з інших ПК надсилатиметься на головний.

Виходячи з заданого завдання, постає питання про вибір топології мережі. Враховуючи надійність та збої в мережі, обирати базову топологію не варіант. Найкращим рішенням є топлогія М-структура. Дана топологія будується на основі найкоротшого дерева та гамільтонового циклу для висячих вершин НКД (вершина, яка зв’язана лише одним ребром з будь-якою іншою вершиною).

Мережа оздоровчого комплексу буде побудована на провідному фізичному середовищі передачі даних.

На рисунку 2.1 продставлено розміщення обладнання – персональних комп’ютерів (ПК) та комутаторів. Чорним кольором позначені основні лінії прокладання кабелької системи, а зеленим пунктиром – додаткові лінії для створення необхідної топології у вигляді М-структури, що необхідно для подальшого імітаційного моделювання.

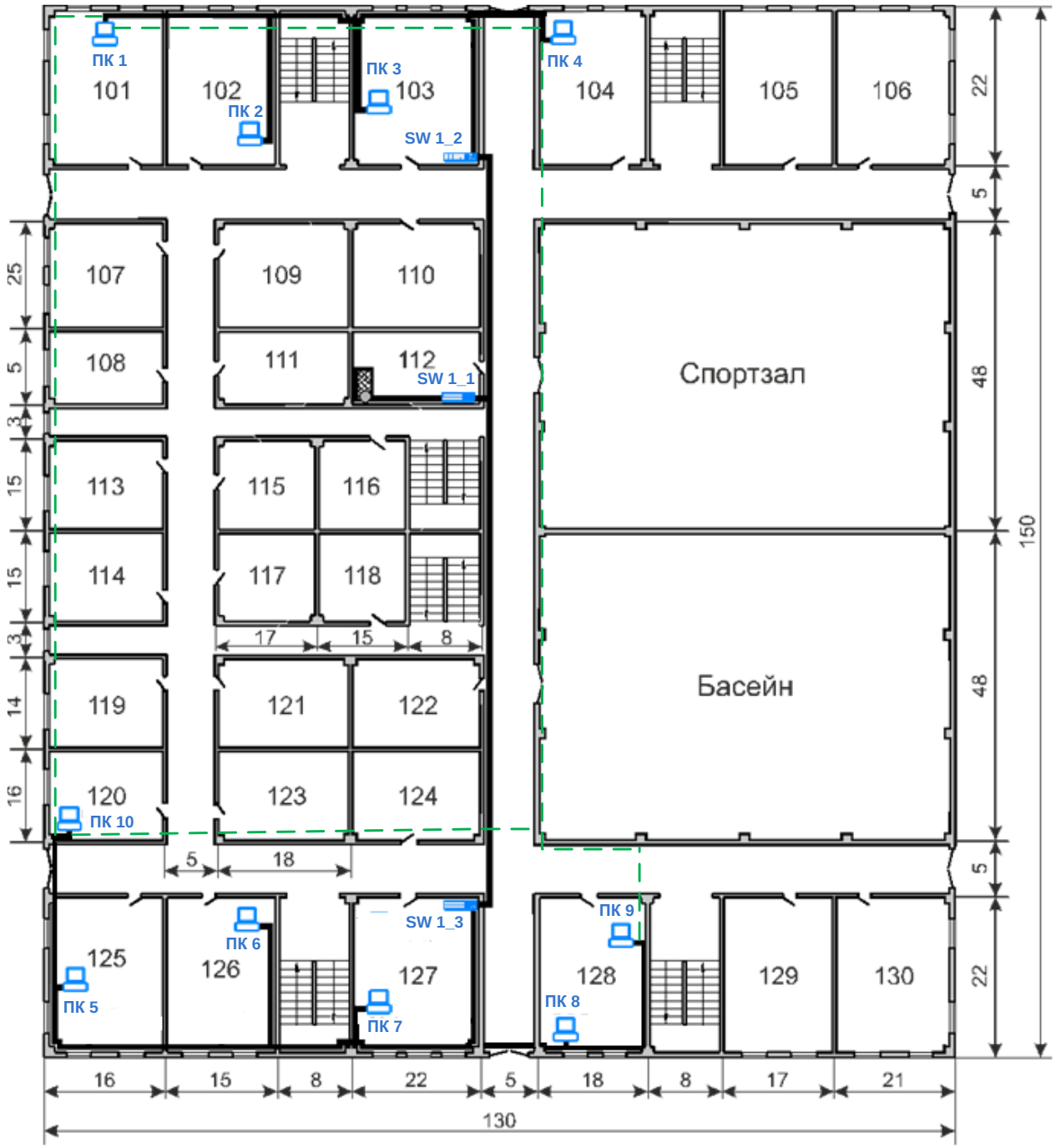

Рисунок 2.1 – Схема прокладання кабельних систем

2.2 Імітаційне моделювання мережі

Відповідно до рисунку 2.1 побудуємо задану топологію мережі у вигляді графу (рисунок 2.2), де ПК – вершини графу, а фізичні лінії зв’зку – ребра.

Для імітаційного моделювання мережі буде використаний програмний пакет Network Simulator, що складається з програми інтерпретатора –

компілятора моделі мережі у вигляді файлу ns.exe і програми візуалізатора - аніматора моделі мережі у вигляді файлу nam.exe.

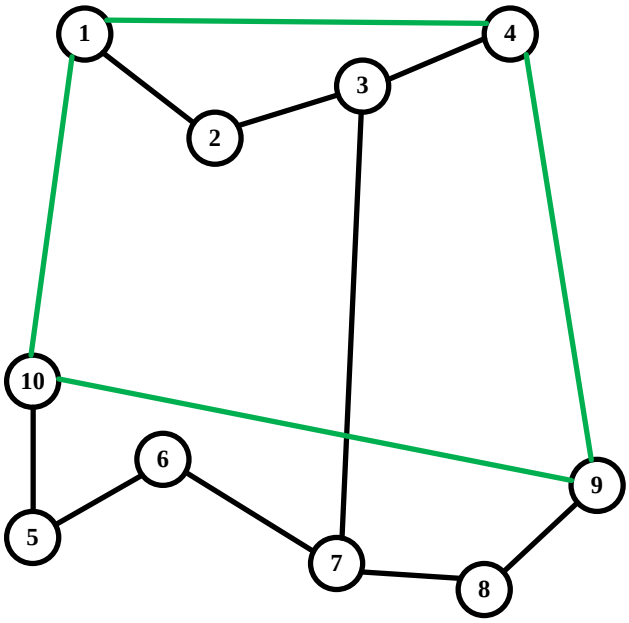

Рисунок 2.2 – Граф мережі

Спочатку створемо в Nam Editor необхідну топологію мережі (рисунок 2.3), що включає 10 вузлів (Node) з відповідними мітками (label) v1, v2,…,v10 – відповідно до значень ПК на схемі, і дуплексні канали (Link) без відмов із чергами типу Drop-Tail (FIFO). Зберегти файл топології з розширенням .ns – Network.ns (рисунок 2.4).

Створюємо в будь-якому текстовому редакторі, що не вносить спецсимволів редагування, проєкт мережі, в якій вузол v1 є центральним - отримувачем пакетів, а інші вузли є відправниками пакетів CBR-агентів.

Зберегти проєкт мережі необхідно у файлі з довільним ім’ям і розширенням .tcl – example4.tcl. в даному файлі прописано, що відправником пакетів є v1 (рисунок 2.5-2.7), на даних скріншотах видно утворення незначних черг.

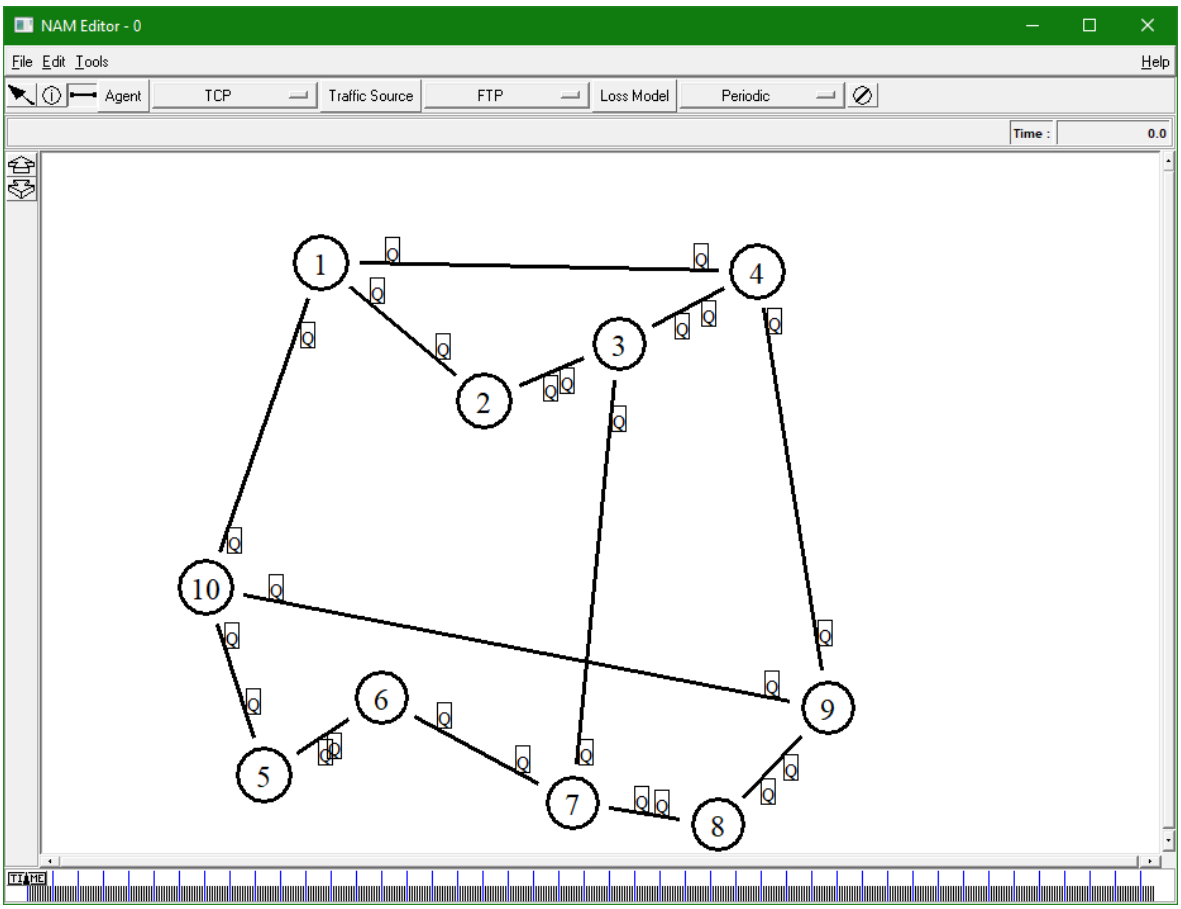

Рисунок 2.3 – Граф мережі в редакторі Nam Editor

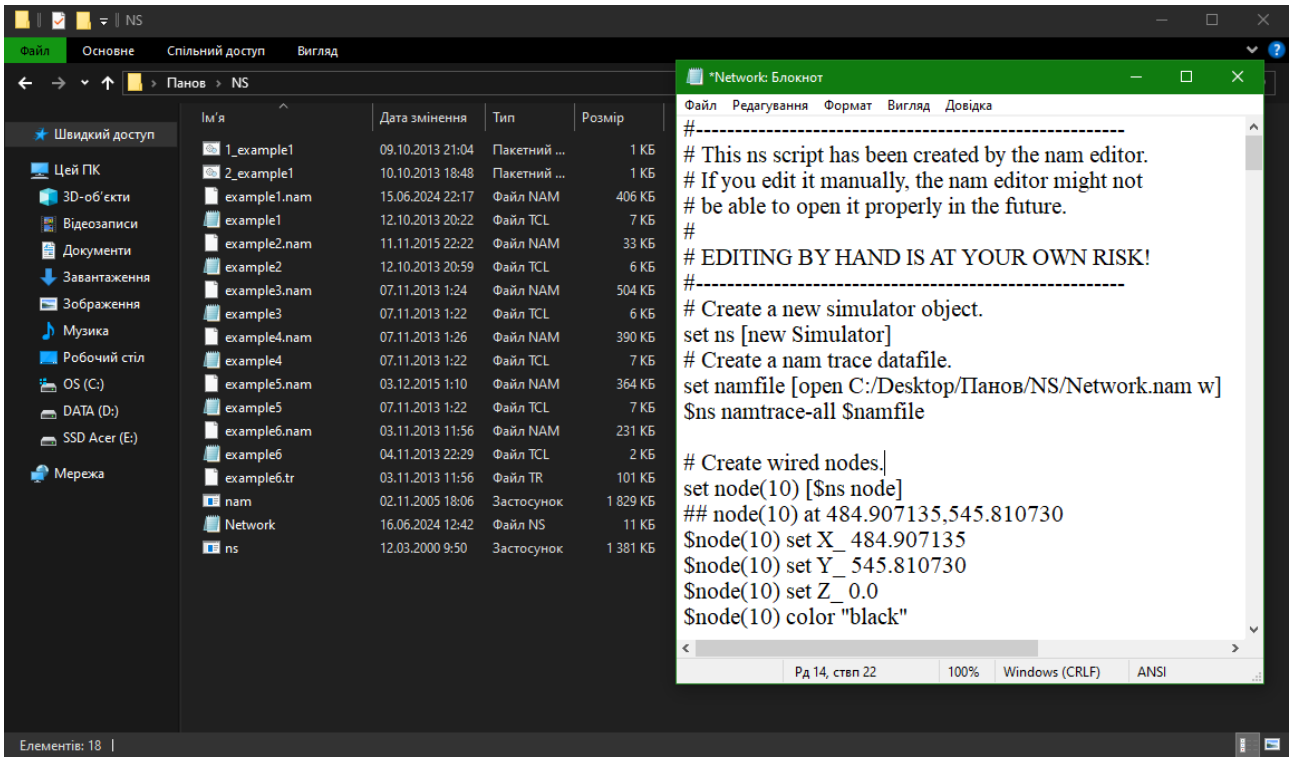

Рисунок 2.4 – Файл Network.ns

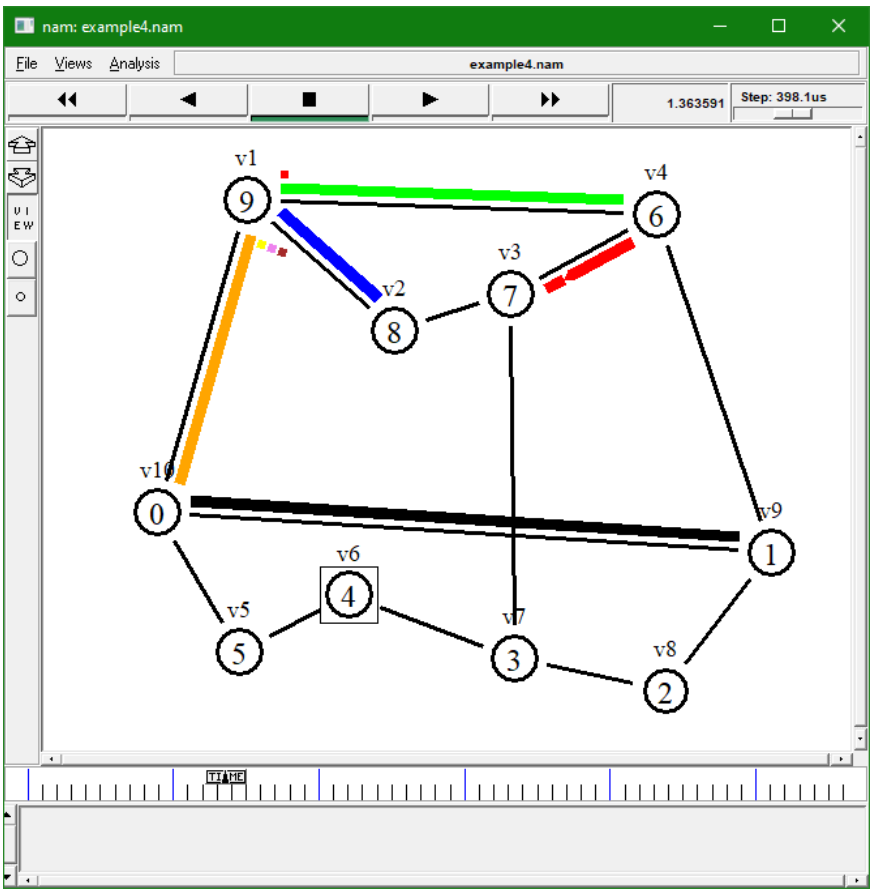

Рисунок 2.5 – Імітаційне моделювання мережі: v1 – відправник

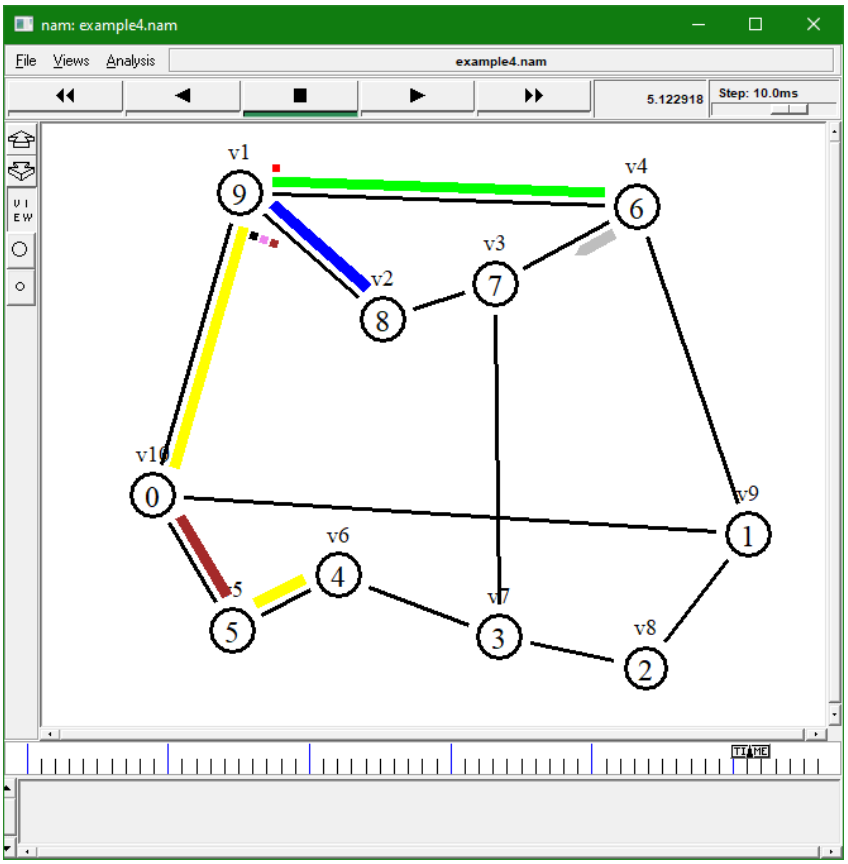

Рисунок 2.6 – Імітаційне моделювання мережі: v1 – відправник

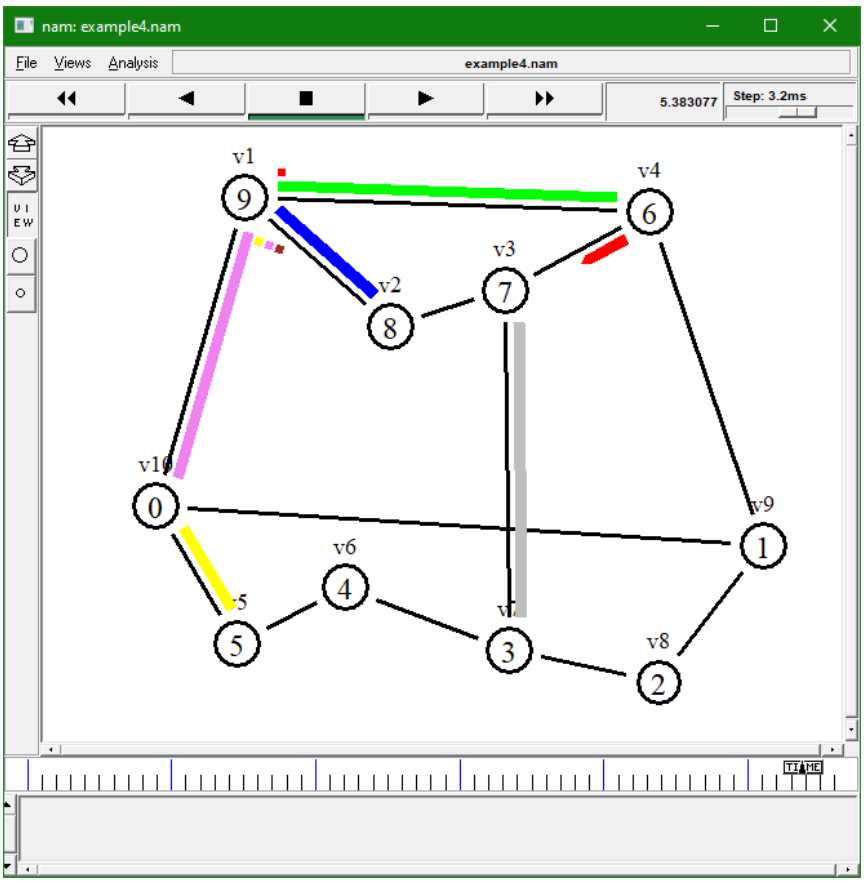

Рисунок 2.7 – Імітаційне моделювання мережі: v1 – відправник

Розглянемо працездатність мережі, коли отримувачем буде v1 (рисунок 2.8-2.10), а решта відправниками. Ця ситуація є більш важливою для розгляду, оскільки навантаження на v1 буде суттєвішим в даному випадку. Також це моделювання покаже, коли саме вузол v1 почне перенавантажуватися та втрачати пекети, відповідно до прописаного коду.

Як видно з імітаційного моделювання, черги в мережі створюються, але нови не значні. Дане моделювання було зроблено також і за наявност несправностей ліній, що дає змогу переглянути відправку пакетів за новоствореними маршрутами. Коли ж канал зв'яку відновлюється, передача пакетів знову відбувається за попереднім маршрутом.

Збої та несправності в роботі ліній зв'язку не минучи для будь-якої розробленої мережі, тому необхідно перевіряти модель мрежі за умов, наближених до реальних. Відмова каналу зв'язку на рисунку 2.11 відображається червоною лінією.

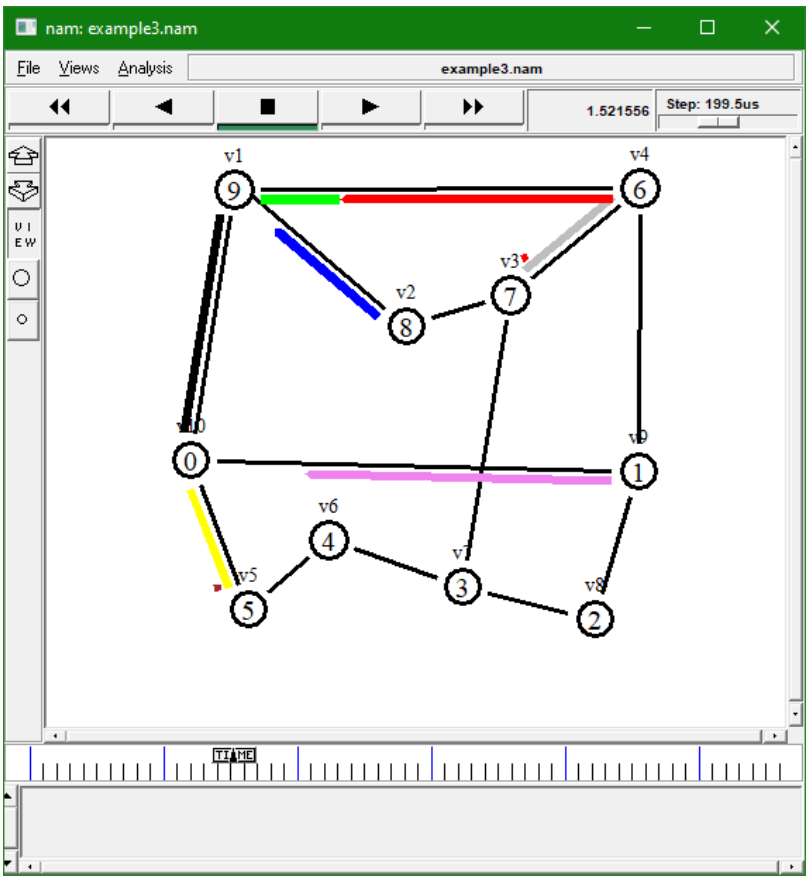

Рисунок 2.8 – Імітаційне моделювання мережі: v1 – отримувач

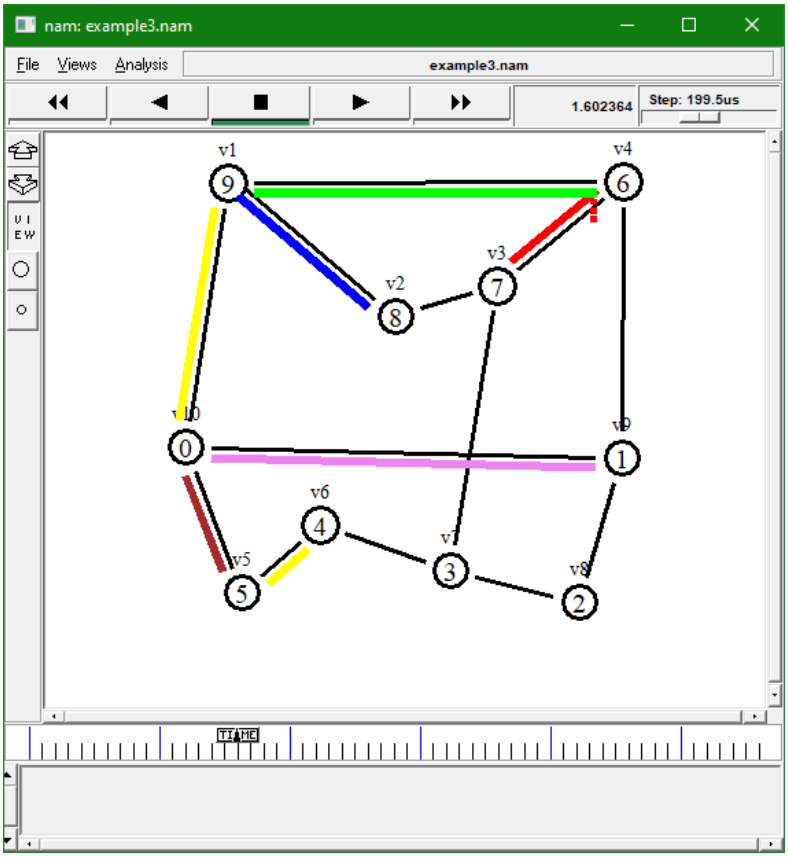

Рисунок 2.9 – Імітаційне моделювання мережі: v1 – отримувач

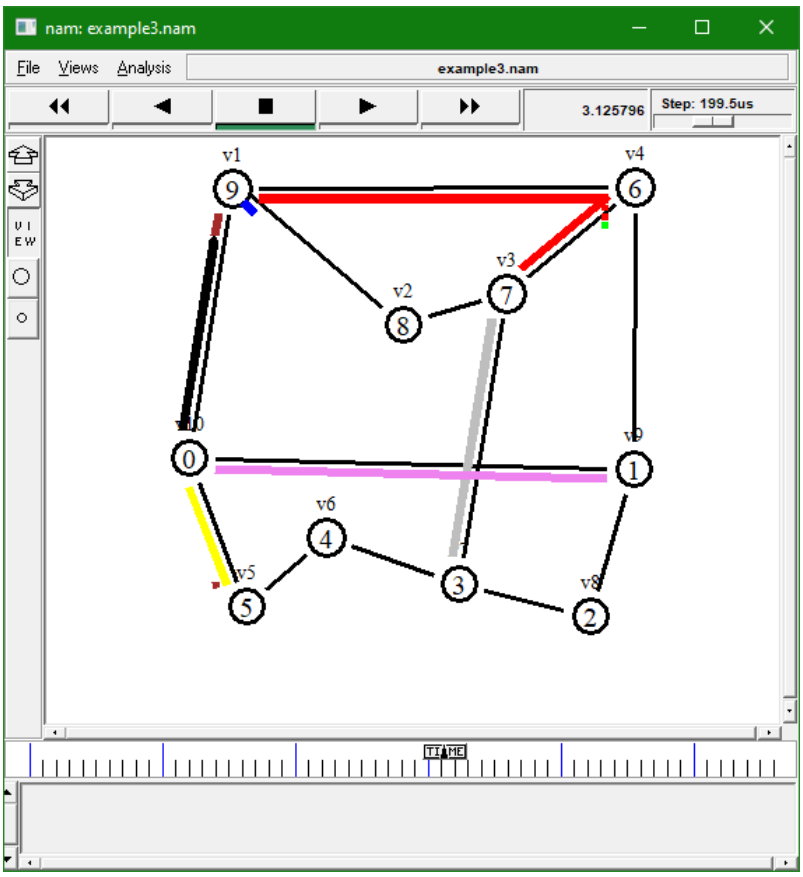

Рисунок 2.10 – Імітаційне моделювання мережі: v1 – отримувач

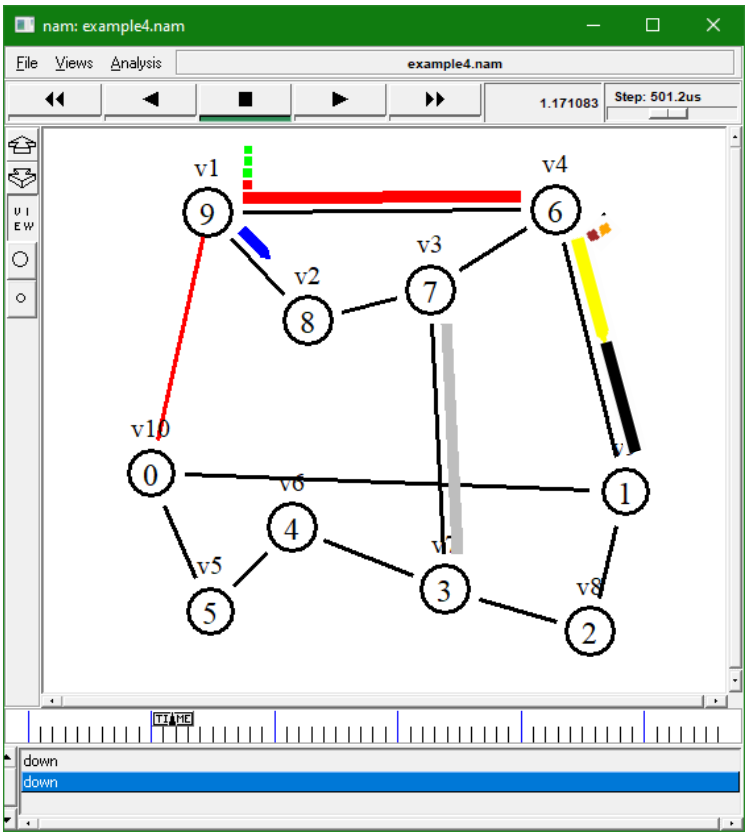

Рисунок 2.11 – Імітаційне моделювання мережі: відмова каналу (1,10)

2.3 Пояснення коду для імітаційного моделювання мережі

ПП NS і NAM є достатньо простими і зручними засобами для вивчення, порівняння і аналізу принципів роботи мереж [2].

ПП NS складається з двох файлів [2]:

– ns.exe – інтерпретатора для мови Object Tool Command Language (Otcl);

– nam.exe – візуалізатора результатів моделювання мережі (network animator) для NS, який дозволяє спостерігати динаміку її функціонування (передачу пакетів по каналам зв'язку, розмір черги пакетів, параметри протоколів та ін.) [2].

Їх запуск виконується з командного рядка будь-якої оболонки (Norton Commander, Volcov Commander, FAR і т.п.) [2]:

ns.exe example.tcl;

nam.exe out.nam;

де example.tcl – текстовий файл з програмою на мові OTcl, який створюється за допомогою будь-якого текстового редактора, що не вносить спецсимволів редагування, і зберігається з розширенням .tcl; out.nam – трасувальний файл, який створюється ns.exe на підставі опису топології та алгоритмів функціонування мережі засобами мови Otcl [2].

Але є ще один спосіб запуску файлів, який і використовувався для зручності роботи. Файл текстового редактору (Блокнот) зберігався з розширенням .bat, для даного випадку потрібно двоє файлів, в кожному з них прописавши по одному з рядків: ns.exe example.tcl та nam.exe out.nam.

На рисисунку 2.12 показано приклад простої імітаційної моделі мережі з комутацією пакетів у ПП Network Simulator (NS), яка включає два вузли мережі і канал зв'язку між ними [2].

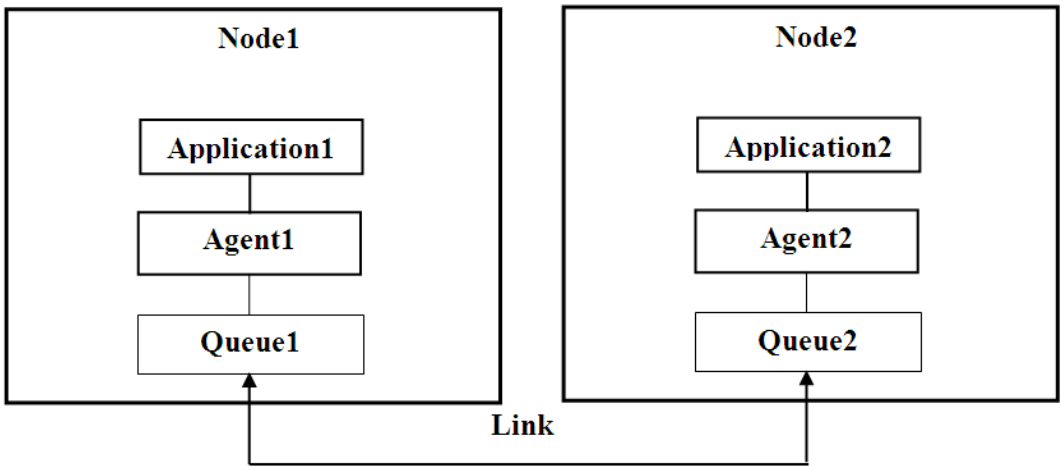

Рисунок 2.12 – Структура взаємодії елементів моделі мережі в ПП NS

Елементи моделі мають таке призначення [2]:

– node (вузол) моделює систему, що має унікальну адресу в мережі [2];

– application (додаток) моделює прикладний процес, що є джерелом трафіку визначеного типу, а дані передаються на нижні рівні [2];

– agent (агент) моделює протокол транспортного рівня, що використовується для формування пакетів, які містять заголовок протоколу і дані від додатка, керує їх потоком, виявляє і відновлює інформаційні пакети і т.д. [2];

– queue (черга) моделює буфер (накопичувач), в якому пакети протоколу транспортного рівня очікують передачі по каналу зв'язку [2];

– link (канал зв'язку) моделює канал передачі з затримкою, інтенсивністю обслуговування (пропускною спроможністю), джерелом помилок [2].

Для прискорення розробки проекту мережі з комутацією пакетів можна використовувати редактор Nam Editor, що запускається через пункт меню «File» → «New Nam Editor…» у додатку Nam Console (рисунок 2.13) аніматора NAM [2].

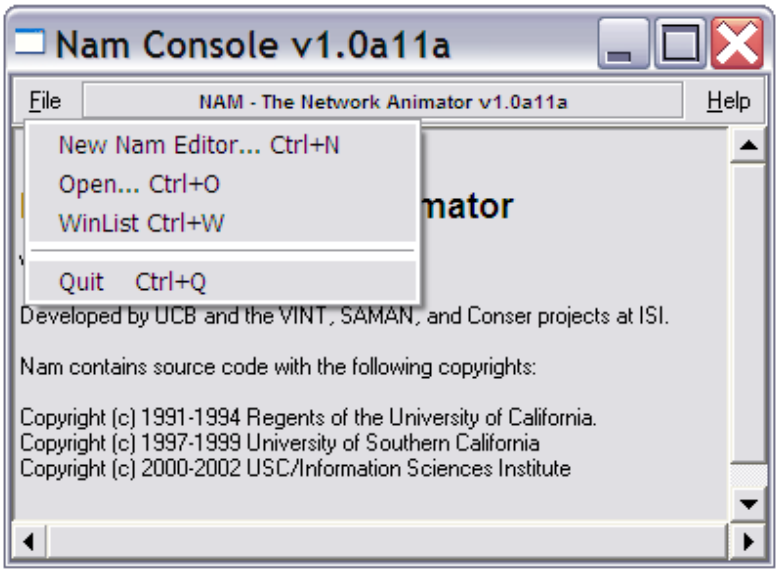

Рисунок 2.13 – Редактор Nam Editor

Редактор Nam Editor дозволяє [2]:

– створювати вузли (Node), для чого потрібно увімкнути режим створення вузлів кнопкою ⓘ на панелі інструментів або використати в меню пункт «Tools»→«Add Node». Після цього можна задавати вузли мережі, натискаючи ліву кнопку миші [2];

– з'єднувати вузли каналами (Link), для чого потрібно увімкнути режим створення каналів кнопкою на панелі інструментів або використати в меню пункт «Tools»→«Add Link». Після цього можна задавати канали мережі, натискаючи ліву кнопку миші спочатку на першому його вузлі, а потім на другому вузлі, і так далі для всіх каналів [2];

– прив'язувати агентів (Agents) до вузлів (Node), для чого потрібно увімкнути режим завдання агентів кнопками Agent Null на панелі інструментів або використати в меню пункт «Tools»→«Agents». Після цього, натискаючи ліву кнопку миші на вузлі (Node), можна задати для нього агентів (Agents), наприклад, отримувача Null [2];

– з'єднувати агентів (Agents) між собою, для чого потрібно увімкнути режим створення з'єднань кнопкою на панелі інструментів або використати в меню пункт «Tools»→«Add Link». Після цього можна задавати з'єднання

агентів (Agents), натискаючи ліву кнопку миші спочатку на першому з них, а потім на другому, і так далі для всіх каналів [2];

– задавати джерела трафіка (Traffic Sources), для чого потрібно увімкнути режим завдання джерел трафіка кнопками 

на панелі інструментів або використати в меню пункт «Tools»→«Traffic Sources». Після цього, натискаючи ліву кнопку миші на агенті (Agents) вузла (Node), можна задати для нього трафік, наприклад, CBR [2];

– задавати модель втрат у каналі (Link Loss Models), для чого потрібно увімкнути режим завдання моделі втрат кнопками 

на панелі інструментів або використати в меню пункт «Tools»→«Link Loss Models». Після цього, натискаючи ліву кнопку миші на каналі (Link), можна задати для нього модель втрат, наприклад, Periodic [2];

– перевизначати властивості об’єктів проекту, для чого потрібно увімкнути режим вибору кнопкою  і виконати подвійне натискання лівої кнопки миші або одинарне натискання правої кнопки миші на об’єкті [2];

– видаляти об’єкти проекту, використовуючи пункт меню «Delete Object» [2];

– - зберігати проект мережі у файлі з заданим ім’ям і розширенням .ns, використовуючи в меню пункт «File»→«Save As…» [2].

Імітаційна модель навіть найпростішої мережі в ПП NS створюється за допомогою рядка: set ns [new Simulator] [2].

Наступною задачею є відкриття файлу out.nam, в якому будуть зберігатися результати дій, які виконуються [2]:

set nf [open out.nam w];

$ns namtrace-all $nf.

Для того, щоб перед посилкою сформованих пакетів виконувався пошук найбільш короткого і прийнятного з погляду пропускної спроможності шляху, використовується динамічна маршрутизація. Однієї з її головних функцій є перенапрямок (ремаршрутизація) пакетів у випадку виникнення в мережі

непередбачених ситуацій (відмова каналу). Наприклад, дистанційно-векторна динамічна маршрутизація може бути задана у вигляді: $ns rtproto DV [2].

Якщо відмови каналів мережі не передбачаються, то зазначений рядок у текст програми не включається [2].

Попередню частину програми, аналогічно опису використовуваних процедур і змінних в мові Pascal, завершує процедура finish, що закриває файл із результатами моделювання і після завершення роботи з ним передає його визуалізатору nam.exe. Одним з можливих варіантів реалізації є запис [2]:

```
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exit 0
}
```

Створення нового вузла (node), візьмимо приклад з файлу example4.tcl:

```
#Створюємо вузли мережі;
set node(10) [$ns node]
## node(10) at 484.907135,545.810730
$node(10) set X_ 484.907135
$node(10) set Y_ 545.810730
$node(10) set Z_ 0.0
$node(10) color "black"
$ns at 0.0 "$node(10) label v10".
```

Перший рядок – створення вузла 10, наступні чотири рядки – координати, які надають інформацію про положення створеної вершини. Наступний рядок – колір, в який зафарбовується вершина і останній рядок – привласнення мітки (label): 10 вершині – v10. Привласнення міток необхідне, оскільки після перенесення інформації (про вузли та лінії зв’язку) з файлу, створеного в Nam Editor, в файл example4.tcl, бо запустивши імітаційну модель, значення в середині вершин не відповідатиме реальним значенням значенням. Тому на

рисунках 2.5-2.10 можна побачити згори саме такі мітки над кожною вершиною.

Клас Link моделює канал зв'язку, що з'єднує два вузли. Для одержання посилання на об'єкт класу Link використовується вираз [2]:

$ns link <вузол> <вузол>.

Методами класу Link є:

– установка ваги гілки для протоколів маршрутизації cost <вага>;

– «включення» каналу up;

– «вимикання» каналу down;

– – повернення показника на елемент черги каналу queue, наприклад:

```
#Визначення каналу зв'язку з обмеженою чергою типу FIFO (DropTail)
$ns duplex-link $n0 $n1 10kb 100ms DropTail
#Отримання показника на канал зв'язку
set l1 [$ns $n0 $n1]
#Отримання показника на чергу
set q1 [$l1 queue].
```

Створення дуплексного каналу:

```
$ns duplex-link $node(8) $node(7) 10.000000Mb 2.000000ms DropTail
$ns duplex-link-op $node(8) $node(7) queuePos 0.5
$ns duplex-link-op $node(8) $node(7) color black
$ns duplex-link-op $node(8) $node(7) orient 171.1deg
# Set Queue Properties for link 8->7
[[$ns link $node(8) $node(7)] queue] set limit_ 20.
```

В першому рядку записується така інформація: duplex-link <вузол> < вузол> <швидкість> <затримка поширення> <тип черги>. DropTail – тип черги в каналі (FIFO – First In, First Out – перший прийшов, першим пішов) [2].

Можна задати перегляд черги для каналу, наприклад, $ns duplex-link-op $n2 $n3 queuePos 0.5, де параметр queuePos 0.5 показує, наскільки рівномірно обслуговуються пакети кожного з джерел [2].

Якщо потрібно переглянути чергу в симплексному каналі, то вираз duplex-link-op заміняється на simplex-link-op [2].

Третій рядок – зафарбовувати лінію зв’язку в певний колір, в даному випадку в чорний. Наступний рядок відноситься до топології і визначається в градусах. Що стосується останнього рядку, тов ньому прописана інформація про розмір буферу для черги відповідно до кожного вузла мережі.

Створення дуплексного каналу також можна представити таким чином: $ns duplex-link $n0 $n2 1Mb 10ms DropTail. У даному випадку вузли з'єднуються дуплексними каналами зв'язку зі швидкістю 1 Мбіт/с, затримкою при передачі пакетів 10 мс (одиниці виміру можна змінити на kb – кбіт/с, b – біт/с, ns – наносекунди, ps – пікосекунди, s – секунди) і чергою за принципом «першим прийшов – першим обслугований» (DropTail).

Після створення вузлів і каналів зв'язку можна задати їхню топологію, наприклад, у вигляді: $ns duplex-link-op $n0 $n2 orient right-down [2].

Оскільки для мереж, що складаються більш ніж з чотирьох вузлів, завдання вручну їхньої топології є громіздким, то доречно використовувати її завдання не в тексті програми, а в самому визуалізаторі nam.exe, використовуючи опцію (кнопку в правому нижньому куті аніматора NAM) re-layout або використовуючи додаток Nam Console аніматора NAM [2].

Формування мережного трафіка можливо в двох режимах: з постійною швидкістю (Constant Bit Rate, CBR) і змінною швидкістю (Variable Bit Rate, VBR) [2].

Один вузол може одночасно формувати пакети різної величини з різною швидкістю. Для формування вузлом заданого типу трафіка, наприклад, CBR для вузла 10, створюється відповідний агент:

```
#Створюємо агента-відправника cbr(10);
set cbr(10) [new Agent/CBR]
#Привласнюємо агента cbr(10) вузлу v1;
$ns attach-agent $node(1) $cbr(10)
#Розфарбуємо пакети, що формуються агентом cbr(10) в колір 9;
```

```
$cbr(10) set fid_ 9
#Задаємо розмір пакетів в байтах (15000);
$cbr(10) set packetSize_ 15000
#Задаємо інтервал між пакетами в секундах (0.09);
$cbr(10) set interval_ 0.09
```

В мові Otcl передбачено кілька типів агентів: одні виеористовуються для формування трафіка (передавальні агенти), а інші – для одержання трафіка (приймаючі агенти). До кожного вузла (отримувача інформації) закріпляється приймаючий «нульовий» агент. Він створюється таким способом (приклад для 5 вершини) [2]:

```
#Створюємо агента-отримувача null(5);
set null(5) [new Agent/Null]
#Привласнюємо агента null(5) вузлу v5;
$ns attach-agent $node(5) $null(5)
```

Для того, щоб пакети, сформовані різними джерелами (агентами), можна було відрізняти один від одного, їх розфарбовують у різні кольори. При цьому агенту привласнюється номер кольору, наприклад, при виборі для них блакитного і фіолетового кольорів, відповідно [2]:

```
$ns color 1 Blue
$ns color 2 Violet
```

Для одержання інших кольорів замість Blue (блакитного) чи Violet (фіолетового) можна використовувати Red (червоний), Green (зелений), Brown (коричневий), Yellow (жовтий), Gray (сірий), Black (чорний), Orange (помаранчовий), Turquoise (бірюзовий) та ін. [2].

Після створення служб формування і прийому пакетів їх необхідно маршрутизувати, тобто направляти пакети, сформовані, наприклад, cbr(0) і cbr(1) до отримувача ("нульовому") агента [2]:

```
$ns connect $cbr(0) $null(0)
$ns connect $cbr(1) $null(0)
```

Після підготовки початкових умов моделювання задається час запуску і зупинки формуючих агентів. Нехай, наприклад, cbr(0) буде запущений через 0,5 с послу початку моделювання і зупинений у момент часу 5,5 с, а cbr(1) буде запущений у 1,0 с і зупинений у 6,0 с [2]:

$ns at 0.5 "$cbr(0) start"

$ns at 1.0 "$cbr(1) start"

$ns at 6.0 "$cbr(1) stop"

$ns at 5.5 "$cbr(0) stop"

При формуванні VBR-трафіка фрагмент програми, що визначає кількість вузлів, канали зв'язку між ними і топологію не відрізняється від аналогічного для мережі з постійною швидкістю трафіка (CBR). Зміни починаються з моменту опису формування трафіку [2].

Завдання передавальних агентів, присвоєння їх вузлам і прив'язка до агентів, що приймають трафік, має вигляд [2]:

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]

set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]

set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

Тут 200 – розмір пакета в байтах; 2s – час формування пакетів (2 сек); 1s – час паузи між пакетами; 100k – швидкість формування (100 кбит/сек).

Початок і закінчення формування задаються, відповідно:

$ns at 10.0 "$source0 start"

$ns at 20.0 "$source0 stop"

Наприкінці програм обов'язково повинен вказуватися час моделювання в секундах, наприклад, 7 с, у вигляді рядка:

$ns at 7.0 "finish", який визначає робочий відрізок часу анімації.

Відмову каналу зв'язку між двома вузлами можна промоделювати як:

$ns rtmodel-at 3.0 down $n1 $n2

$ns rtmodel-at 5.0 up $n1 $n2

Саме ці дії також були застосовані для розробленої імітаційної моделі, бо дають можливість дослідити принцип проходження пакетів за наявності несправностей деяких каналів.

У цьому випадку канал зв'язку між вершинами 1 і 2 відмовить (рядок 1) у момент часу 3,0 с, а потім відновиться в момент часу 5.0 с.

Текст програми закінчується обов'язковим рядком:

$ns run, який запускає компілятор.

Клас ErrorModel моделює джерела помилок у каналах зв'язку. В клас ErrorModel входять такі підкласи.

Uniform – для випадкової втрати елементів з рівномірним розподілом у діапазоні [0, 1]. При створенні об'єктів підкласу передається частка елементів, що втрачаються, і елементів, що спотворюється, (pkt або byte). Наприклад, для визначення втрати 1 відсотка всіх елементів [2]:

set err1 [new ErrorModel/Uniform 0.01 pkt],

а для визначення втрати 1 відсотка всіх октетів (байтів):

set err1 [new ErrorModel/Uniform 0.01 byte]

#Інсталяція джерела помилок у каналі зв'язку після черги заявок

$ns link-lossmodel $err1 $n0 $n1

MultiState – для джерела помилок у виді об'єкта з дискретними станами, заданого матрицею переходів між ними і тривалостями перебування в кожному стані, наприклад, ErrorModel/Uniform [2].

#Визначення моделей втрат в кожному стані

set tmp [new ErrorModel/Uniform 0 pkt]

set tmp1 [new ErrorModel/Uniform .9 pkt]

set tmp2 [new ErrorModel/Uniform .5 pkt]

#Визначення кількості станів

set m_nstates 3

#Формування масиву станів

set m_states [list $tmp $tmp1 $tmp2]

#Визначення початкового стану

```
set m_nstart [lindex $m_states 0]
#Визначення тривалості перебування в кожнім стані
set m_periods [list 0 .0075 .00375]
#Формування матриці переходів між станами
set m_transmx { { 0.95 0.05 0 } { 0   0   1 } { 1   0   0 } }
set m_trunit pkt
set m_sttype time
```

#Визначення змінної типу ErrorModel/MultiState джерела помилок, а як параметри при створенні передаються визначені раніше змінні:

```
set err1 [new ErrorModel/MultiState $m_states $m_periods $m_transmx $m_trunit $m_sttype $m_nstates $m_nstart]
#Інсталяція джерела помилок в каналі зв'язку після черги заявок
$ns link-lossmodel $err1 $n0 $n1.
```

ВИСНОВКИ

Основним інструментом аналізу інфокомунікаційних мереж є імітаційне моделювання в комп'ютерних програмах-симуляторах без застосування реального обладнання. На етапі побудови моделі виникає питання про вибір інструментарію. При цьому важливими потребами для створення ефективної моделі є: детальна реалізація протоколів мереж; можливість написання і підключення власних модулів; можливість зміни параметрів моделювання під час проведення експериментів; платформна незалежність; розвинений графічний інтерфейс; ціна продукту.

До імітаційного моделювання прибігають, коли: дорого або неможливо експериментувати на реальному об’єкті; неможливо побудувати аналітичну модель, тому що в системі є час, причинні зв'язки, наслідки, нелінійності, стохастичні (випадкові) змінні; необхідно зімітувати поведінку системи в часі.

В кваліфікаційні роботі наведено характеристики кількох популярних програм імітаційного моделювання різного класу – від простих програм до потужних систем, які включають бібліотеки більшості наявних на ринку комунікаційних пристроїв та забезпечують можливість значного ступеня автоматизації дослідження цієї мережі.

В роботі було змодельовано мережу для оздоровчого комплесу. Дане імітаційне моделювання допомогло наглядно переконатися в роботі мережі за різних налаштувань мережі. При великих навантаженнях на мережу виникаю черги та можливі втрати пакетів, але дане питання вирішується налаштуванням мережі з вищими числовими позниками параметрів для передачі пакетів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Квєтний Р.Н., Богач І.В., Бойко О.Р. Софина О.Ю. Комп’ютерне моделювання систем та процесів. Методи обчислень. [Електронний ресурс] – Режим доступу: https://web.posibnyky.vntu.edu.ua/fksa/2kvetnyj_komp'yuterne_modelyuvannya_system_procesiv/t1/zm1..htm (дата звернення: 19.05.2024)

2. Пустовойтов П.Є. Сучасні програмні засоби оптимізації та моделювання інфокомунікаційних мереж. – Харків: ХНУРЕ, 2017. – 119 с. (дата звернення: 19.05.2024)

3. Зеляновський М.Ю., Алхіхі Мухамад. Засоби для моделювання спеціалізованих та сенсорних мереж бездротового доступу: симулятори роботи комп’ютерних мереж NS-2 та NS-3. [Електронний ресурс] – Режим доступу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/21633/21-ZelianovskijNEW.PDF?sequence=4 (дата звернення: 22.05.2024)

4. Уривський Л.О., Мошинська А.В., Осипчук С.О. Імітаційне моделювання систем і процесів у телекомунікаціях: навч. посіб.. Київ: КПІ ім. Ігоря Сікорського, 2022. – 202 с. (дата звернення: 22.05.2024)

5. Дубовой В.М., Юхимчук М.С. Імітіційне моделювання в системі SCILAB/XCOS: Е-посібник. ІРВЦ ВНТУ. – 2018. [Електронний ресурс] – Режим доступу: https://web.posibnyky.vntu.edu.ua/fksa/10dubovyj_imitacijne_modelyuvannya_v_systemi_Scilab-Xcos/ (дата звернення: 20.05.2024)

6. Жерновий Ю.В. Імітаційне моделювання систем масового обслуговування: Практикум. Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. – 307 с. (дата звернення: 19.05.2024)

7. Буртняк І.В. Імітаційне моделювання: методичні рекомендації для студентів спеціальності економічна кібернетика. Івано-Франківськ : ПНУ, 2019. – 97с. (дата звернення: 20.05.2024)