МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Поляруш Богдан Сергійович

Програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Доренський Олександр Павлович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Полярушу Богдану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349*

керівник роботи *Доренський Олександр Павлович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Поляруш Б.С. Програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи визначення пропускної здатності мережі на основі методології RFC6349.

Метою розробки є програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349.

Результат роботи – програмна реалізація системи визначення пропускної здатності мережі на основі методології RFC6349.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі RAD Studio Delphi.

Ключові слова: комп'ютерна інженерія, RFC6349

ABSTRACT

Poliarush B.S. Network bandwidth determination software based on RFC6349 methodology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of definition of bandwidth of a network on the basis of methodology RFC6349 is developed.

The purpose of the development is the software of the network bandwidth determination system based on the RFC6349 methodology.

The result is a software implementation of a network bandwidth determination system based on the RFC6349 methodology.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment of RAD Studio Delphi.

Keywords: computer engineering, RFC6349

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	Програмне забезпечення
ЦОД	–	Центр обробки даних
FC	–	Fibre Channel
IT	–	Інформаційні технології
ITO	–	Internetwork Throughput Option, пропускна здатність в мережі
LAN	–	Локальна мережа
SAN	–	Мережі зберігання даних
SLA	–	Рівень сервісу
TCP	–	Transmission Control Protocol, протокол керування передачею
UDP	–	Протокол дейтаграмм користувача
WAN	–	Глобальна мережа

ВСТУП

Актуальність теми. Ви є IT-фахівцем, відповідальним за мережу корпусів університету або підприємства? Або ж ви працюєте в компанії, яка є постачальником послуг, і ваше завдання полягає в тому, щоб забезпечити мінімальну пропускну здатність між постачальником і віддаленою компанією або житловим будинком? Вам теж потрібні докази того, що пропускна здатність між двома точками усередині цієї мережі може реально досягати заявленого значення. На щастя, існують інструменти, які допоможуть вам у виконанні цього завдання.

Найчастіше для перевірки пропускної здатності використовується відповідний метод тестування. При проведенні типового тестування пропускної здатності трафік з обраною вами швидкістю й протягом заданого періоду часу відправляється з одного мережного пристрою на інший. Пристрій, що ухвалює, рахує кількість отриманих під час тестування кадрів. Потім здійснюється обчислення швидкості приймання, яка також називається пропускною здатністю.

У випадку якщо при передачі не був загублений жоден кадр, пропускна здатність буде дорівнює швидкості передачі. Однак якщо між двома точками тестування утворюється так зване пляшкове горлечко, то кадри будуть загублені й пропускна здатність виявиться нижче, чим швидкість передачі. Щоб довідатися максимальну пропускну здатність лінії, почніть із максимальної теоретичної швидкості передачі й поступово знижуйте швидкість, поки на пристрої, що ухвалює, більше не буде загублений жоден кадр.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем визначення пропускної здатності мережі на основі методології RFC6349.

– Дослідження системи визначення пропускної здатності мережі на основі методології RFC6349.

– Програмна реалізація системи визначення пропускної здатності мережі на основі методології RFC6349.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі визначення пропускної здатності мережі на основі методології RFC6349.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Тестування пропускної здатності – це двостороння перевірка: один пристрій є основним (сервером), а інше – віддаленим (клієнтом). Основний тестовий модуль відправляє трафік, а віддалений – ухвалює й вимірює його. Тестовими пристроями можуть бути спеціально призначені для цієї мети прилади або портативні комп'ютери з відповідним програмним забезпеченням. Тестування пропускної здатності є однаково ефективним при тестуванні локальних (LAN) і глобальних (WAN) мереж. При тестуванні пропускної здатності проводиться перевірка всієї лінії мережі, що полягає з декількох компонентів: кінцевих пристроїв і всього мережного устаткування, яке перебуває між ними (концентраторів, комутаторів, точок доступу й маршрутизаторів). Кожний компонент містить у собі кілька складових, таких як мережні інтерфейсні карти й порти, материнські плати й операційні системи. Зміна кожної із цих складових спричинить зміна пропускної здатності мережі.

На пропускну здатність також може впливати тип переданого трафіку. Обробка комутаторами й маршрутизаторами кадрів малого розміру (наприклад, 64 байт) займає більше часу, що приводить до зниження пропускної здатності, якщо дані пристрої не встигають обробляти всю інформацію. Уміст кадрів (усі одиниці, усі нулі, випадкова послідовність біт) також може впливати на пропускну здатність. Це пов'язане з відмінностями в схематику й алгоритмах, використовуваних різними мережними адаптерами, комутаторами й маршрутизаторами. На максимальну пропускну здатність може впливати й тривалість тестування. Мережне устаткування, що володіє великою обчислювальною потужністю й розширеними буферами, довше зможе справлятися з високою швидкістю трафіку, чому менш “здат ні” пристрої. І

нарешті, завершує список причин впливу на пропускну здатність протокол передачі. Швидкість протоколу керування передачею (TCP, Transmission Control Protocol) є більш низкою, чому швидкість протоколу дейтаграмм користувача (UDP), оскільки TCP використовує послідовні повідомлення про успішне приймання даних і при необхідності повторно передає пакети.

1.2 Область застосування

Перш ніж починати тестування пропускної здатності, рекомендуємо вам вибрати сервісний порт. Порт за замовчуванням залежить від тестового застосунку, з яким ви працюєте. Перевірте, щоб ваш брандмауер не блокував обраний вами порт і щоб і основний, і віддалений модуль використовували той самий порт. Якщо тестування пропускної здатності проводиться в активній мережі, вам слід пам'ятати, що результати тестування свідчать про пропускну здатність у конкретний момент часу. В активно працюючій мережі випадкові колізії й загублені кадри – нормальне явище. У зв'язку зі звичайною активністю мережі результати вашого тестування при кожному вимірі можуть суттєво різнитися. Для встановлення базисного рівня продуктивності активної мережі ви можете виконувати тестування пропускної здатності з регулярним тимчасовим інтервалом, щоб задокументувати поведінка мережі в змінних умовах завантаження.

Тестування пропускної здатності також є додатковою функцією аналізатора. Це функція перевірки пропускної здатності в мережі (ITO, Internetwork Throughput Option). Використовуючи два аналізатори Etherscope у якості кінцевих пристроїв, ми можемо тестувати пропускну здатність зі швидкістю до 1000 Мбіт/с. На відміну від більшості тестових приладів, що вимірюють пропускну здатність тільки в одному напрямку (від основного модуля до віддаленого), аналізатор виконує дві функції: і основного, і віддаленого тестового інструмента. Це допомагає провести двостороннє тестування, тому ми

можемо виміряти пропускну здатність в обох напрямках, і нам не прийдеться міняти або повторно налаштовувати кінцеві пристрої. Економія часу при цьому помітна неозброєним оком.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Network and Service Companion (NSC-100)

Компактний пристрій для техніків і підрядників, що займаються монтажем, активацією й пошуком несправностей у широкосмугових житлових, корпоративних і міських мережах. Універсальний засіб порівняльного виміру працездатності мережі, перевірки функціонування й розподілу мережних служб відповідно до передових методів

Network and Service Companion (NSC-100) – нова категорія компактних портативних пристроїв, у яких кілька тестових інтерфейсів (PON, Ethernet, Wi-Fi) сполучаються з концепцією застосунку VIAVI Onecheck, що дозволяє фахівцям з монтажу й настроюванню, незалежно від рівня їх кваліфікації, виконувати перевірку й тестування високошвидкісних (гігабітних і більш швидких) мереж швидше й з більшою зручністю.

Єдиний ефективний інструмент, який дозволяє технікам працювати в різних мережних середовищах і умовах використання, фіксуючи працездатність і якість послуг мереж, а також швидко й точно виявляючи будь -які проблеми менше, чим за хвилину.

Особливості:

– Можливість для техніків швидко, зручно й комплексно перевіряти активацію послуг і виявляти несправності при підключенні по різних інтерфейсах.

– Можливість тестувати не тільки доставку сигналу PON-мережі до кінцевої точки, але й розподіл послуг, використовуючи тестові інтерфейси PON, Ethernet і Wi-Fi.

– Досить одного натискання кнопки для запуску застосунку Onecheck, що робить перевірку мережі й тестування служб зручним і швидким процесом.

– Забезпечує точність порівняльного тестування на рівні точки входу й подальшого розподілу в межах будинку або приміщення із чітким і швидким виявленням проблем.

– Гарантує дотримання передових методів для скорочення кількості скарг, дзвінків у службу підтримки й відтоку клієнтів.

– Управляється безпосередньо через застосунок VIAVI Mobile Tech або використовується разом з OneExpert (ONX) для розширення можливостей.

Сфери застосування:

– Впровадження послуг, їх уведення в експлуатацію й пошук несправностей.

– Мережі доступу.

– PON – для FTTH і 5G x-haul.

– CATV – для DOCSIS 3.1.

– Телекому – для xDSL і Gfast.

– Metro/Ethernet.

– Корпоративна мережа й ЦОД.

– Послуги для бізнесу.

– Магістральна мережа бездротової мережі – 4G і 5G.

Основні характеристики:

– Міцна конструкція без екрана, для використання в польових умовах.

– Тестовий інтерфейс PON з емуляцією ONU/ONT.

– Тестові інтерфейси Ethernet до 10G.

– Підтримка режиму закольцовування Ethernet L2/L3 для тестів Y.1564 і RFC-2544 на швидкостях до 10 Гбіт/с.

– VIAVI Fusion (централізована система тестування) використовує NSC у якості дистанційно керованих тест-агентів.

– Повноцінний інтерфейс тестування Wi-Fi: антена Wi-Fi 3x3 з підтримкою 2,4 і 5 Ггц; Wi-Fi Expert відображає покриття, шум у каналі, тривалість передачі й пропускну здатність.

– Автоматична функція Onecheck забезпечує виконання техніком усіх необхідних і обов'язкових перевірок при знаходженні на місці.

– Кілька рівнів тестування пропускної здатності гігабітних і більш швидких мереж (Ookla Speedtest, Truespeed RFC-6349, FTP Speedcheck) для оцінки сервісу стосовно необхідних еталонних показників.

– Тести швидкості сервісу для всіх тестових інтерфейсів.

– Job Manager допомагає технікам у керуванні завданнями, результатами тестів і звітністю.

– Вбудовані можливості формування й відправлення звітів через хмару (Stratasync) або по електронній пошті (через смартфон/планшет).

Stratasync – хмарний застосунок, який забезпечує керування ресурсами, настроюваннями й тестовими даними вимірювальних приладів VIAVI, а також наявність останнього ПЗ й опцій. Stratasync Core безкоштовно підтримує платформу OneExpert . Розширені можливості керування даними пропонуються у версії Stratasync Plus.

Мультисервісний модуль MSAM для платформи MTS-6000A/8000

Мультисервісний модуль MTS-6000A/8000 (MSAM) як і раніше зберігає домінуюче положення на ринку пристроїв для установки й обслуговування мереж (RFC2544, RFC6349, ITU/T-Y.1564) завдяки продуманому дизайну й апаратній платформі, що відповідає вимогам конвергентних Ip-мереж, що й одержують усе більше поширення опорних і периферійних мереж 10G. Цей модуль також придасться при установці й діагностиці мереж зберігання даних (SAN), включаючи 8G Fibre Channel (FC), які використовуються для мінімізації дублювання елементів інфраструктури ЦОД, скорочення затримок і організації резервного копіювання.

Ця флагманська платформа MTS як і раніше орієнтована на обслуговування всього життєвого циклу мереж від установки (RFC2544,ITU/T-Y.1564) до діагностики й обслуговування (RFC6349). MTS-6000A/8000 MSAM – це передова багатопортова платформа 10G, розроблена для технічних фахівців, що працюють у польових умовах і в центральних офісах (CO), яким потрібно перевіряти різноманітні технології передачі даних від TDM/PDH до 10G OTN, включаючи 8G FC, і забезпе чувати виконання застережених вимог до рівня сервісу (SLA) відповідно до галузевих рекомендацій і стандартами (RFC2544, RFC6349, ITU/T-Y.1564).

Переваги:

– Максимальна окупність інвестицій завдяки модульній платформі, розрахованої на підтримку майбутніх технологій і інтерфейсів, що й допускає модернізацію в польових умовах.

– Передова багатопортова модульна платформа 10G для установки мереж у польових умовах, поглибленої діагностики й лабораторного апробування мереж (RFC2544, RFC6349, ITU/T-Y.1564).

– Гарантія задоволеності кінцевих замовників за рахунок поглибленого тестування до рівня застосунків (Truespeed™ по RFC 6349)), голосу (VoIP) і відео (IPTV) на рівні застосунків.

– Підвищення ефективності життєвого циклу обслуговування мережі й керування мережею за рахунок наявності інтегрованого інструментарію для установки, поглибленої діагностики й аналізу (RFC2544, RFC6349, ITU/T-Y.1564).

– Прискорення установки й діагностики мереж завдяки наявності автоматизованих тестерів J-Complete, спроектованих на основі передової практики, що й використовують відтворені методики й процедури, що дають легені в інтерпретації результати.

Застосування:

– Тестування й діагностика конвергентних мереж Ethernet/IP на швидкостях від 10 Мбіт/с до 10 Гбіт/с із можливістю захвата й аналізу пакетів.

– Тестування SLA Ethernet/IP рівнів 1-3 за допомогою автоматичного поглибленого тестера RFC 2544/Samcomplete згідно ITU-T Y.1564.

– Тестування TDM/PDH у мережах SONET/SDH на швидкостях від OC-3/STM-1 до OC-192/STM-64, включаючи вимір кількісних показників порушень надання послуг і захват заголовків маршруту (POH) за допомогою тригерів.

– Установка й обслуговування мереж OTN до 11,1 Гбіт/с із підтримкою ODU-0 для клієнтських інтерфейсів Ethernet/IP.

– Підтримка двопортових інтерфейсів 8G FC (поряд з 1/2/4/10 Гбіт/с) для установки й обслуговування SAN і каналів з малої затримкою.

Модуль CSAM для платформи MTS-6000A/8000

Компактність і широкі можливості тестування мережі з підтримкою всіх швидкостей аж до 100 Гбіт/с.

Призначений для установки в устаткування MTS для тестування, дозволяючи створювати компактні й портативні застосунки для тестування 100-гігабітних мереж як у польових умовах, так і в центральному офісі.

Компактний модуль CSAM забезпечує гнучке тестування мереж 100 Гбіт/с. Тестуйте мережі на швидкостях від 10 Мбіт/с до 118 Гбіт/с, використовуючи портативний інструмент тестування, що дозволяє перевіряти кожний інтерфейс у центральному офісі. Унікальність CSAM – у тому, що один модуль підтримує 100GE LR4,SR10 і SR4.

Основні переваги:

– Портативний модуль для тестування на швидкостях від 10 Мбіт/с до 118 Гбіт/с із повною підтримкою інтерфейсів Ethernet, OTN, SONET/SDH і Fiber Channel.

– Функція внутрішнього тестування оптоволокна в експлуатаційних умовах забезпечує справну роботу модулів CFP2, CFP4, QSFP+ і QSF P28. Підтримка всіх інтерфейсів 100GE, включаючи LR4, SR10 і SR4.

– Економія часу завдяки використанню останніх галузевих тестів Samcomplete™ RFC 2544 і Y.1564 на відповідність Ethernet-служб угодам про рівень обслуговування (SLA), включаючи високоточні виміри затримки в комбінації з попереднім тестуванням Ethernet Quickcheck.

– Висока швидкість і ефективність тестування активації сервісів у мережах OTN. Є унікальний автоматичний сценарій перевірки мереж OTN.

– Усуває складні в діагностиці проблеми площини керування Ethernet завдяки інтегрованому тестуванню рівня Layer 2 на прозорість.

Сфери застосування:

– тестування Ethernet, OTN, Fiber Channel і SONET/SDH.

– Перевірка на дотримання угоди про рівень обслуговування Ethernet і OTN.

Цільова аудиторія:

– Фахівці з обслуговування міських, магістральних і підводних кабелів опорної мережі операторів фіксованому й мобільному зв'язку, що займаються розгортанням і забезпеченням працездатності устаткування від 10 М біт/с до 100 Гбіт/с.

– Фахівці з обслуговування ЦОД і організацій, користувачі, що займаються розгортанням і забезпеченням сигналів до 100 Гбіт/с.

Network and Service Companion (NSC-100)

– Компактний пристрій для техніків і підрядників, що займаються монтажем, активацією й пошуком несправностей у широкосмугових житлових, корпоративних і міських мережах. Універсальний засіб порівняльного виміру працездатності мережі, перевірки функціонування й розподілу мережних служб відповідно до передових методів.

Fusion

Система моніторингу й тестування продуктивності мережі на основі віртуальних пробників.

Модуль TEM

Виміру параметрів часу й синхронізації на базі платформи MTS -5800 в експлуатаційних умовах

Розроблений для застосування в експлуатаційних умовах модуль розширення системи синхронізації (TEM) допомагає операторам мережі підтримувати точну синхронізацію, забезпечуючи лідируючі в галузі показники точності при вимірі параметрів часу й синхронізації.

Швидка, динамічна адаптація й координація мереж радіодоступа (RAN) буде вимагати більш точної синхронізації від однієї генерації до наступної. Модуль розширення системи синхронізації компанії VIAVI Solutions допомагає операторам мережі підтримувати точну синхронізацію, що забезпечує бездоганна якість обслуговування клієнтів.

Платформа MTS-5800 і розроблений для застосування в експлуатаційних умовах модуль TEM забезпечують лабораторну точність вимірів параметрів часу й синхронізації. Особливостями цього модуля є наявність у сучасного 72-канального приймача GNSS входу антени й надзвичайно точних мініатюрних рубідієвих атомних годин (MAC) для забезпечення вимірів з точністю до наносекунди навіть при відсутності сигналу від супутника й при роботі модуля в режимі очікування.

Особливості

MTS 5800 і модуль TEM дозволяють:

– Вимірювати однобічну затримку для виявлення асиметричних затримок у мережі.

– Робити виміри збою відліку часу через PTP, що дозволяє впевниться в стабільності й точності ваших граничних годин PTP.

– Точно вимірювати PDV PTP для здачі в експлуатацію або діагностики мереж, що не підтримують протокол PTP.

– Правильно встановлювати антени GNSS, вимірюючи рівень сигналу супутника й усуваючи збої в роботі антени.

– Підвищувати точність вихідних сигналів 1 PPS устаткування за допомогою аналізу вандера 1 PPS.

– Вимірювати джиттер і вандер на T1 і E1.

Переваги:

– Бездоганна якість обслуговування клієнтів; глибока синхронізація бездротових мереж забезпечує менша кількість скинутих викликів і збільшує швидкість на стороні абонента.

– Зручність; TEM компактний дизайн, що має, є доповненням для мультипротокольного тестера MTS 5800, усуваючи необхідність мати при собі другий тестовий набір для синхронізації.

– Точність; постійна наносекундна точність навіть у режимі очікування при наявності виняткового приймання із супутника.

Основні характеристики:

– Вимір затримки по Ethernet і однобічної затримки (OWD) IP пакетів.

– Підтвердження синхронізації по частоті, фазі й часу в експлуатаційних умовах з точністю, наближеної до лабораторної.

– Підвищення точності установки й поліпшення якості роботи антен GNSS.

– Підтримка декількох сузір'їв GNSS, включаючи GPS, GLONASS, Beidou, SBAS і QZSS.

– Швидкий і точний захват сигналу супутника за допомогою сучасного 72-канального приймача GNSS.

– Підтримка декількох входів 1 PPS і 10 МГц і синхровиходов.

– Виміру параметрів 1588v2 (PTP), включаючи виміру PDV і часу з наносекундною точністю.

– Емуляція ведучого пристрою PTP (PRTC).

– Виміру однобічної затримки в каналах 1/10 GE з наносекундною точністю.

MTS 5800-100G

Тестування мережі й оптоволокна за допомогою багатофункціонального портативного приладу.

Портативний мережний тестер MTS 5800-100G – це той самий інструмент, який необхідний технічним фахівцям і інженерам для організації, настроювання й технічного обслуговування мереж. Він підтримує як традиційні, так і передові технології мережних застосунків, включаючи застосунки для тестування базових і міських мереж, мереж центру обробки даних і мереж надання комерційних послуг.

Самий компактний у галузі портативний тестер із двома портами 100 Гбіт забезпечує можливість тестування мережі протягом усього часу її використання, включаючи перевірку оптоволокна, активацію сервісів, проведення діагностики неполадок і технічного обслуговування. Тестер 5800-100G оснащений новітніми інтерфейсами, у тому числі SFP/SFP+/SFP28 і QSFP+/QSFP28/CFP4, що надовго забезпечить актуальність його використання в обслуговуванні мереж, що постійно розбудовуються, без зниження рівня задоволеності замовників. Завдяки таким функціям тестування, як внутрішня перевірка оптоволокна, вимір і декодування лінійної швидкості Ethernet-З'єднання й перевірка мереж OTN, фахівці одержали можливість більш точно й швидко тестувати мережі.

Переваги:

– Багатофункціональний портативний тестер із двома портами 100 Гбіт спрощує комплексну діагностику мереж.

– Тестер оптимально підходить для роботи в експлуатаційних умовах. Він оснащений сенсорним екраном з функцією «multitouch», підтримує роботу із заданих сценаріїв і надає результати тестування у форматі «справно/несправне».

– У тестері реалізовані передові технології з можливістю відтворення застосовуваних методик і процесів.

– Прилад прискорює тестування оптоволокна, активацію сервісів Ethernet і діагностику мереж.

Сфери застосування:

– Тестування й налагодження комбінованих мереж Ethernet/IP зі швидкістю передачі даних від 10 Мбіт/с до 100 Гбіт/с, застосовуваних у центрах обробки даних, опорних і міських мережах.

– Визначення характеристик, діагностика й усунення несправностей оптоволокна.

– Розгортання й технічне обслуговування мереж OTN, мереж раніше використовуваних стандартів SONET/SDH і Dsx/PDH.

– Визначення характеристик, перевірка, діагностика й усунення несправностей (включаючи синхронізацію) волокна в мобільних і магістральних мережах. Сумісність зі стандартом 5G.

Основні характеристики:

– Підтримка комплексного тестування швидкості передачі даних у діапазоні від стандарту Dsx/PDH (1,5/2 Мбіт) до стандарту OTU4 (112 Гбіт).

– Найшвидший у галузі тест активації Ethernet-сервісів RFC 2544 і Y.1564 Samcomplete™ заощаджує час. Мережна затримка виміряється з наносекундною точністю. Крім того, підтримується тест RFC 6349 Truespeed.

– Функція внутрішнього тестування оптоволокна в експлуатаційних умовах забезпечує справну роботу модулів QSFP+/QSFP28 і CFP4.

– Висока швидкість і ефективність тестування активації сервісів у мережах OTN. Є автоматичний сценарій перевірки мереж OTN.

– Сумісність із оптичними рефлектометрами VIAVI серії 4100 і модулями COSA з функцією Smart Link Mapper™, мікроскопами для тестування оптики й вимірниками оптичної потужності.

– Можливість перевірки синхронізації й міток часу за допомогою модуля TEM (Timing Expansion Module).

Портативний тестер мережі MTS-5800

Даний тестер є самим компактним у галузі тестером мережі 10 Гб. Протягом усього терміну служби може забезпечувати якісне тестування, включаючи визначення характеристик оптоволокна, активацію послуг, усунення неполадок і технічне обслуговування.

Портативний тестер мережі MTS_5800 – це той інструмент, який необхідний і фахівцям з обслуговування мереж, і мережним інженерам для установки, пуско-налагодження й технічного обслуговування мереж. Він підтримує як застарілі, так і передові технології, необхідні для обслуговування різних мережних застосунків, включаючи установку мереж масштабу міста/базових мереж, передачу стільникового трафіку й комерційні послуги. Самий компактний у галузі портативний прилад протягом усього терміну служби може забезпечувати якісне тестування, включаючи визначення характеристик оптоволокна, активацію послуг, усунення неполадок і технічне обслуговування. Такі розширені функції тестування мереж Ethernet, як Truespeed для RFC 6349, J-Profiler™, захват/розшифрування зі швидкістю, відповідній до середовища передачі даних, а також автоматизована система J-Mentor, дозволяють технічним фахівцям на місцях тестувати мережі швидше й точніше, чим коли-або колись.

Переваги:

– Це портативний багатофункціональний пристрій тестування дубльованих мереж 10 Гб спрощує процес тестування з використанням декількох технологій.

– Оптимізоване для використання в експлуатаційних умовах завдяки мультисенсорному екрану, зашифрованим робочим процесам і чіткому структурованому відображенню результатів.

– У тестері реалізовані передові технології з можливістю відтворення застосовуваних методик і процесів.

– Прискорює визначення характеристик оптоволокна, активацію послуги Ethernet, виявлення й усунення несправностей.

– Надає фахівцям базових станцій можливість легко встановити й підтримувати прямі й зворотні з'єднання.

– Дозволяє уникати дорогих і небезпечних підйомів на вишки.

Характеристики:

– Підтримка повністю завантажених TDM/PDH для дубльованої мережі 10 Гб Ethernet, SONET, SDH, Fibre Channel і OTN.

– Автоматизована вдосконалена технологія тестування RFC 2544 і Samcomplete для ITU-T Y.1564.

– Інтегровані засоби тестування імпульсної потужності MEF 34 і тестування пропускної здатності RFC 6349 Truespeed™TCP.

– Інтегровані засоби тестування хронометрування/синхронізації, включаючи PTP/1588v2, Synce, Wander і тестування однобічної затримки One Way Delay.

– Інтегровані засоби тестування прямих з'єднань CPRI/OBSAI Рівня 1/Рівня 2 і засобу емуляції основної смуги частот і трансіверів.

– Версії з одним і двома портами.

– Сумісність із оптичними рефлектометрами VIAVI серії 4100 і Smart Link Mapper™, мікроскопами для тесту оптики й вимірниками оптичної потужності.

– Емуляція BBU.

– Rfocpri.

Сфери застосування

– Визначення характеристик, перевірка, діагностика й усунення несправностей волокна в мобільних мережах і зворотних з'єднаннях базових станцій.

– Тестування конвергентних мереж Ethernet/IP, діагностика й усунення несправностей інтерфейсів мереж від 10 Мб/с до 10 Гб.

– Визначення характеристик, діагностика й усунення несправностей оптоволокна.

– Розгортання й технічне обслуговування мереж OTN і застарілих мереж SONET/SDH і TDM/PDH.

– Впровадження й обслуговування технологій бездротової синхронізації, таких як PTP/1588v2 і Synce.

– Тестування дистанційних радіоблоків (RRH) на бездротовій базовій станції.

– Тестування працездатності дистанційних радіоблоків RRH без підйомів на вишку за допомогою підключень CPRI і емуляції BBU.

– Визначення пасивної интермодуляції й виявлення перешкод через точку доступу для тестування оптоволокна (Rfocpri).

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000

виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64 -bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи визначення пропускної здатності мережі на основі методології RFC6349.

В процесі розробки кваліфікаційної бакалаврської роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для виміру пропускної здатності каналів зв’язку розподіленої мережі розроблений застосунок у якому виміри виконуються автоматично й тільки в періоди найменшої активності внутрішніх користувачів, що забезпечує високу вірогідність результатів і дозволяє використовувати їх для обґрунтування претензій провайдерам мережних послуг. Пропускна здатність каналів виміряється на рівні TCP.

Ви – IT-менеджер великої компанії, відповідальний за роботу розподіленої корпоративної мережі. У центрі мережі (наприклад, у центральному офісі) розташований ЦОД, доступ до якого з віддалених офісів здійснюється по орендованих каналах зв'язку. Для керування такою мережею необхідно вирішити, як мінімум, три завдання:

1. **Організувати постійний контроль доступності каналів зв’язку (availability).** Якщо якийсь канал упаде, ви зможете довідатися про це до того, як користувачі звернуться в Service Desk. Крім того, непогано знати, дотримують чи NSP (Network Service Providers, Провайдери Мережних Послуг) свої зобов'язання по доступності каналів зв’язку. Це завдання порівняно нескладне. Для її застосунку можна використовувати будь-яку систему моніторингу, яка із центру по ICMP буде пінгувати устаткування віддалених офісів.

Звичайно в SLA із провайдерами мережних послуг (NSP, Network Service Provider) сформульовані вимоги тільки до доступності й фізичної швидкості орендованих каналів. Фізична швидкість постійна. Тому єдине, що можна перевіряти, це доступність каналів.

2. **Організувати постійний контроль якості передачі даних по каналах зв'язку (jitter, delay, packet loss).** Контроль необхідний для швидкої діагностики кореневих причин інцидентів (скарг користувачів), зокрема, для визначення, винувата чи в них мережа. Це трохи більш складне завдання, і її можна вирішувати різними способами:

– Якщо каналоутворююче устаткування підтримує технологію IP SLA (наприклад, устаткування Cisco Systems), кращим способом є впровадження системи моніторингу, що підтримує дану технологію; див. Паспорт якості Ip-Каналу.

– Якщо використовується устаткування, що не підтримує IP SLA, тобто кілька шляхів: установити в мережі спеціальні апаратні зонди (самий правильний, але й самий дорогий варіант); обмежитися моніторингом якості роботи каналоутворюючого устаткування, наприклад, утилізації портів, числа помилок і т.п.; впровадити Навантажувальний Моніторинг Мережі.

3. **Періодично проводити аудит пропускної здатності каналів зв'язку на рівні TCP.** Такий аудит необхідний для контролів якості послуг NSP. Вас цікавить пропускна здатність на рівні TCP, тому що більшість критично важливих бізнес-застосунків працює саме по TCP. При цьому вам потрібна достовірна інформація, отримана на основі великої кількості вимірів (представницької вибірки). Це дозволить при розмові з NSP аргументовано обґрунтовувати свої претензії. У розділі ми розглянемо, як це завдання вирішується методом Навантажувального Моніторингу Мережі, підтримуваного (у різному ступені) усіма продуктами сімейства Prolan SLA-ON (Адміністратор, Аналітик, Експерт), у тому числі безкоштовним продуктом Qutester Plus.

Застосунок не сертифікований, тому отримані з його допомогою результати при дозволі юридичних суперечок правової сили не мають.

Чому не Iperf або Chariot?

Для виміру пропускної здатності мережі часто використовують утиліти типу Iperf, Chariot і т.п. Вони ідеально підходять для разових вимірів пропускної

здатності мережі, наприклад, на етапі її пуско-налагодження (або приймання-здачі), але незастосовні для проведення аудита. Це пояснюється, як мінімум, трьома причинами:

1. При проведенні аудита найцікавіша пропускна здатність мережі в робочий час. Саме в цей час опорна мережа провайдера найбільш завантажена, і її пропускна здатність може знижуватися. Але саме в цей час мережа найбільше активно використовується внутрішніми користувачами. Тому, якщо ви будете вимірювати пропускну здатність мережі за допомогою Iperf або Chariot у робочий час, те, по-перше, результати вимірів будуть дуже не точні , по-друге, це негативно відіб'ється на роботі користувачів.

2. Для одержання репрезентативної вибірки кожний канал потрібно виміряти не менш 50 раз. Чим більше вимірів, тем достовірніше результат.
Виконувати виміру вручну – дуже трудомістке завдання. Можна написати скрипт, який буде запускати тестування автоматично. Але це не таке просте завдання. Скрипт повинен уміти аналізувати готовність сервера, перезапускатися при збоях і багато чого іншого.

3. Крім того, якщо з'ясується, що пропускна здатність орендованих каналів гірше очікувань, і ви захочете висунути аргументовані претензії NSP, вам буде потрібно прив'язати результати всіх вимірів вчасно. Тільки в цьому випадку NSP зможе зіставити результати ваших вимірів з даними своєї системи моніторингу, тільки так він зможе визначити, «хто винуватий», і спробувати усунути вузьке місце. При використанні Iperf і Chariot прив'язку вчасно потрібно робити вручну. Це складно, трудомістко й велика ймовірність помилки.

Юридичних претензій до NSP ви пред'явити, швидше за все, не зможете, тому що для цього необхідно, по-перше, щоб вимірник був сертифікований, по-друге, щоб в SLA були прописані гарантії на пропускну здатність каналів на рівні TCP, що дуже малоймовірно.

Тому для проведення аудита пропускної здатності мережі потрібні інші застосунки. Прикладом такого застосунку є Навантажувальний Моніторинг

Мережі. Цей вимір ефективної пропускної здатності мережі (network throughput), виконуване методом Керованої Генерації TCP-трафіку із заданими параметрами.

Як це працює

У ЦОДі встановлюється система моніторингу, що включає Зонд (їх може бути трохи) і консоль керування. У віддалених офісах установлюються відповідачі під Windows або Linux. Для проведення Навантажувального Моніторингу Мережі використовується Тест пропускної здатності мережі на рівні TCP.

Тест пропускної здатності мережі на рівні TCP – це VB-скрипт, виконуваний на Зонді. Зонд – комп'ютер під керуванням будь-якої версії Windows, на якому виконується служба SLA-ON Probe. Робота Тесту заснована на генерації UDP і TCP-трафіку між Зондом і Відповідачами. UDP використовується тільки для моніторингу доступності Відповідачів: Зонд із заданою періодичністю пінгує Відповідачі по UDP. Для навантаження каналів і виміру їх пропускної здатності використовується TCP. Відповідач – це служба Linux або Windows, яка може працювати на серверах або вбудовуватися в активне устаткування, наприклад, у роутери.

У параметрах настроювання Тесту можна задавати:

1. Режим генерації трафіку:

– Послідовно один канал за іншим або всі канали одночасно. У першому режимі генерація здійснюється по черзі між Зондом і кожним Відповідачем. У другому режимі генерація здійснюється між Зондом і всіма Відповідачами одночасно.

– Моніторинг Мережі й Навантажувальне Тестування. У першому випадку, між Зондом і Відповідачем із заданою періодичністю передається масив даних фіксованого розміру (від 1 МБ до 100 МБ). У другому випадку між Зондом і Відповідачем (або Відповідачами) протягом певного періоду часу виконується передача даних з максимально можливою інтенсивністю.

2. Розмір блоку даних, якими здійснюється обмін між Зондом і Відповідачами.

3. Напрямок передачі даних:

- тільки від Зонда до Відповідача;
- тільки від Відповідача до Зонда;
- одночасно в обох напрямках;
- спочатку в одному напрямку, потім в іншому напрямку.

4. Розклад генерації:

- с заданою періодичністю (наприклад, кожні 15 хвилин);
- у заданий час (наприклад, щодня в 0:15, 2:45 і т.п.).

5. Додаткові параметри:

- використовувати або не використовувати при передачі даних алгоритм Найгла;
- контролювати або не контролювати доступність Відповідачів по UDP;
- включити (виключити) керування генерацією трафіку; якщо режим включений, те скільки часу очікувати настання умов генерації, перш ніж відкласти генерацію (див. нижче).

Ключовою перевагою Навантажувального Моніторингу Мережі є підтримка Керованої Генерації Трафіку. Для цього в Зонд додається спеціальна програма Регулювальник, яка постійно стежить, можна або не можна в цей момент виконувати генерацію трафіку. Генерація трафіку почнеться тільки в тому випадку, якщо Регулювальник говорить «Можна». Якщо в процесі генерації трафіку Регулювальник скаже « Не можна», то генерація припиняється. У найпростішому випадку, Регулювальником може бути небагато модифікований Професійний тест оцінки «здоров'я» комутаторів, що підтримують MIB-II, що виконує, зокрема, моніторинг завантаженості портів каналоутворюючого маршрутизатора.

Припустимо, тестується канал зв'язку, підключений до 6-му порту маршрутизатора. При цьому Зонд підключений до 1-му порту, а користувачі – до

4-му порту. Припустимо, Тест пропускної здатності мережі повинен з 9:00 до 20:00 щогодини передавати 10 Мбайт даних від Відповідача до Зонда.

Генератор трафіку постійно стежить за сигналом Регулювальника й почне генерацію трафіку тільки в тому випадку, якщо Регулювальник говорить «Можна». А це відбудеться тільки в тому випадку, якщо утилізація порту 4 буде менше певного значення, наприклад, 5%. Якщо в той час, коли повинна початися генерація трафіку, Регулювальник говорить « Не можна», то Генератор буде чекати певний час. Якщо протягом цього часу він так і не дочекається сигналу «Можна» (зниження утилізації до 5%), то генерація трафіку буде відкладена до наступного години. Почавши генерацію трафіку, Генератор продовжує контролювати сигнал Регулювальника, і якщо він побачить сигнал « Не можна» (утилізацію порту 4 вище 5%), те відразу припиняє генерацію, фіксує конфлікт, і анулює результати даного виміру.

Регулювальник може працювати в тлі, і, таким чином, заздалегідь знати, можна або не можна в цей момент часу виконувати генерацію трафіку (навіть якщо його не запитують), а може запускатися на вимогу (якщо перевірка умов виконується швидко). Перший режим переважніше. Умови видачі сигналів «Можна» і « Не можна» можуть бути самими різними (не тільки утилізація портів). Це може бути, наприклад, число активних підключень до бази даних або число активних користувачів якогось бізнесов-застосунку. При необхідності Регулювальника можна відключати. Регулювальник не входить до складу Тесту (тому що практично завжди вимагає кастомизації), а поставляється додатково на возмездні основі.

3.2 Розробка структурної схеми

Кожний з нас задавався питанням, чому фільм або файл гойдається зі швидкістю нижче, чим швидкість, заявіена провайдером. Наприклад, швидкість 50 Мбіт/сек, а швидкість у програмі для завантаження не більш 5-6 Мбіт/сек.

Багато застосунків працюють на основі транспортного протоколу TCP, наприклад, HTTP, HTTPS, FTP і т.д. І не багато знають, що на продуктивність застосунку в цілому впливає не тільки канальний (MAC) або мережний рівень (IP), але й транспортний (TCP або UDP), та й усі інші. Тому що користувацький комп'ютер або сервер можуть і вміють гальмувати. Для застосунку цієї проблеми й беручи до уваги, що корпоративні застосунків практично все працюють на основі TCP, в 2010 році зібралися представники VIAVI Solutions, Bell Canada і Deutsche Telecom і створили рекомендацію RFC 6349.

RFC6349 – це нова методологія тестування пропускної здатності каналів зв'язку на транспортному рівні, яка опублікована Internet Engineering Task Force (IETF) і рекомендується до використання як застосунок до тестів на MAC і IP рівні (стандарт ITU-T Y.1564), що дозволить підвищити продуктивність корпоративних застосунків і орендованих каналів зв'язку.

У рекомендації RFC6349 описані параметри, які впливають на пропускну здатність каналу зв'язку, як провести необхідні виміри в польових умовах і інтерпретувати результати:

– визначення максимального розміру корисного блоку даних одного пакета (MTU) відповідно до RFC 4821 на всьому шляху проходження, що дозволить визначити максимальний розмір пакета без фрагментації корисного навантаження;

– вимір базової кругової затримки й мінімальної пропускної здатності для визначення оптимального розміру вікна (TCP window size) і розрахунків TCP Bandwidth-Delay Product (BDP);

– виконання декількох вимірів пропускної здатності мережі на транспортному рівні, щоб максимально заповнити трубу з виявленими параметрами на попередніх етапах.

Опишемо кожний із цих етапів більш докладно.

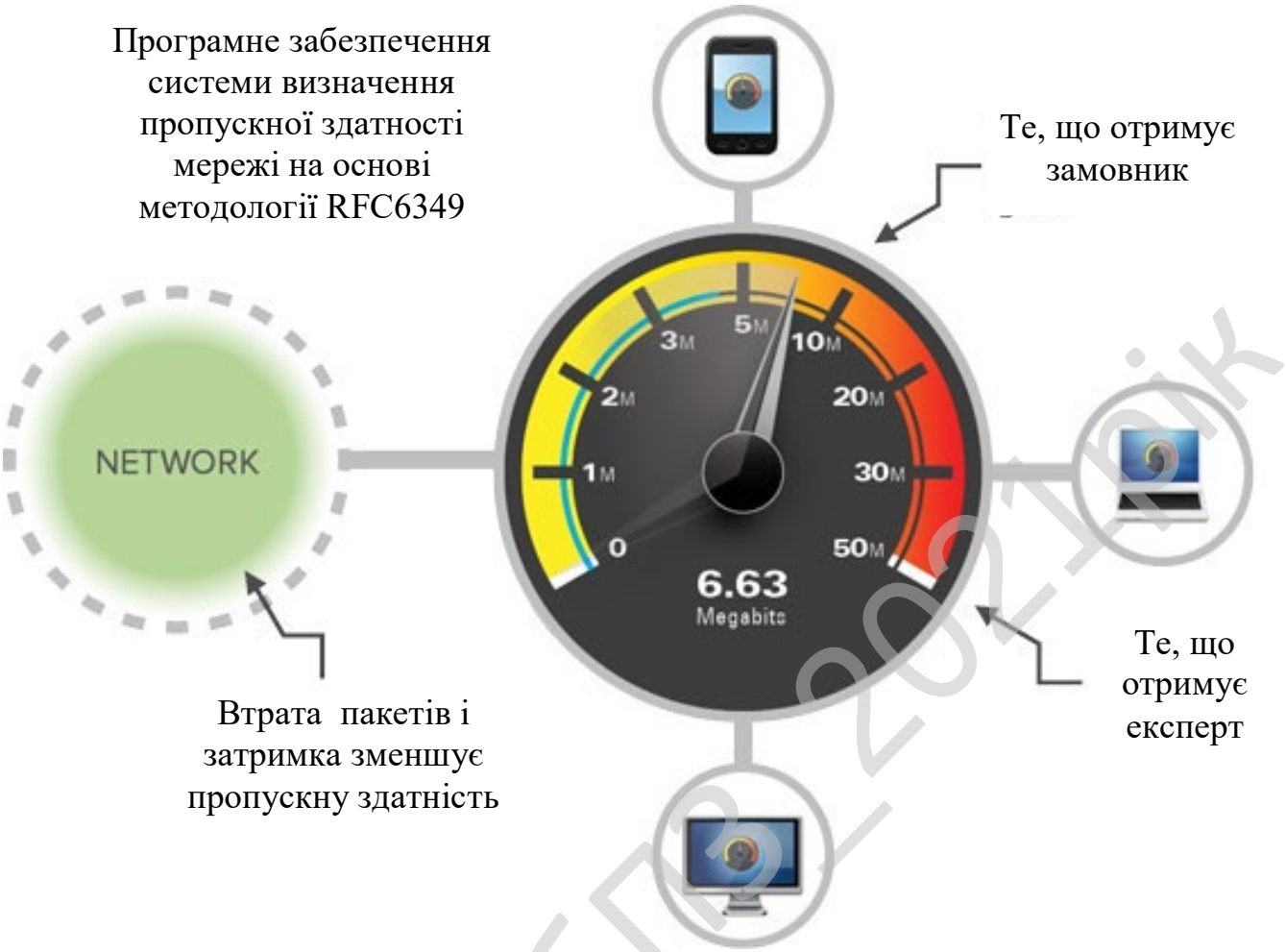

Рисунок 3.1 – Структурна схема системи

Визначення MTU по RFC 4821

Протокол TCP використовує технологію визначення максимального розміру корисного блоку даних одного пакета (PMTUD) на основі протоколу Internet Control Message Protocol (ICMP). Кожний з нас може перевірити MTU на шляху проходження до якого або ресурсу. Для цього досить виконати Ping з додатковими параметрами:

Якщо мережному пристрою необхідно передати пакет без його фрагментації, то в IP заголовку є спеціальне поле, яке дозволяє або забороняє використовувати фрагментацію (DF).

Якщо на шляху проходження такого пакета з'явиться маршрутизатор з параметром MTU менше, чим довжина переданого нефрагментуємого пакета, то

маршрутизатор його відкине й повинен відправити ICMP повідомлення про це із вказівкою параметра MTU для повторної передачі й розбивка пакета відправником до необхідного розміру.

Але в реальних мережах мережні фахівці блокують передачу ICMP, що робить механізм підстроювання MTU неможливим. Тому RFC6349 рекомендує протестувати канали зв'язку й визначити MTU для всього шляхи проходження пакетів відповідно до RFC 4821, який може працювати з або без підтримки протоколу ICMP – Packetization Layer Path MTU Discovery (PLPMTUD).

Базова кругова затримка й мінімальна пропускна здатність

Базова кругова затримка (baseline round-trip time, RTT) – це час, необхідне для передачі повного TCP сегмента: від першого біта до останнього. Даний параметр виміряється в момент мінімального навантаження на канали зв'язку. Якщо тестування виконується в період активного навантаження, то виміру треба зробити кілька раз і вибрати мінімальний час для того щоб прийняти його як базове. Як можна виміряти базову кругову затримку:

– підключити два вимірювальні прилади по обидві сторони каналу зв’язку й виконати виміру відповідно до RFC5357;

– виконати захват трафіків за допомогою аналізатора протоколів і обчислити час на етапі встановлення TCP з'єднання – потрійне рукостискання між пакетами SYN і SYN-ACK;

– виконати PING, але слід розуміти, що ICMP має найнижчий пріоритет з погляду TOS, DSCP.

Мінімальна пропускна здатність (bottleneck bandwidth, BB) виміряється в обох напрямках, особливо якщо канали асиметричні й у різні періоди дня, щоб побрати саме мінімальне значення. Для цього можна виконати класичний тест відповідно до RFC2544 або Y.1564.

Далі ці два параметри використовуються для розрахунків мінімально необхідного розміру вікна TCP (TCP receive window, RWND) і розміру буфера для оптимальної продуктивності на TCP рівні:

BDP (bits) = RTT (sec) * BB (bps),

Min TCP RWND = BDP(bits) / 8

Після визначення мінімального розміру TCP вікна ми можемо розрахувати пропускну здатність на TCP рівні:

TCP Throughput = TCP RWND * 8/ RTT

Наприклад, ми орендуємо канал 100 Мбіт/сек і кругова затримка становить 10 мсек, а розмір вікна налаштований на 64 Кбайт, тоді пропускна здатність складе:

TCP Throughput = 64000*8/0,01 = 51,2 Мбіт/сек

Для того щоб розрахувати максимально можливу пропускну здатність необхідно скористатися формулою:

Maximum TCP Throughput = (MTU – 40) in Bytes * 8 bits * max FPS

FPS – Frame per second на фізичному рівні. Для Ethernet розраховуєтьс я в такий спосіб:

FPS = (Speed (Mbps) / (1538 Bytes * 8 bits))

Як виходить 1538 байта = 1500 (MTU) + 14 (заголовок Ethernet) + 4 (CRC32) + 12 (Inter-Frame Gap) + 7 (преамбула) + 1 (Start of Frame Delimiter).

Таким чином, для каналу 100 Мбіт/сек максимальне значення складе:

FPS = (100/ (1538*8)) = 8127 біт/сек

Maximum TCP Throughput = (1500 – 40) * 8* 8127 = 94,9 Мбіт/сек

Таким чином, можна зробити вивід, що для збільшення пропускної здатності ми можемо або скоротити кругову затримку або побільшати розмір TCP вікна, тобто передати більше даних до одержання підтвердження про його доставку. У більшості випадків розмір буфера для відправлення даних і вікно одержання не оптимально налаштовані на комп'ютерах і це вимагає уваги для збільшення продуктивності саме застосунків при передачі по орендованих вами каналам зв'язку.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Представимо опис розробленого програмного комплексу для визначення пропускної здатності мережі на основі методології RFC6349 на основі аналітичного й імітаційного моделювання..

Завдання визначення пропускної здатності мережі на основі методології RFC6349 складається у визначенні пропускних здатностей каналу зв'язку із урахуванням неоднорідності трафіку, різноманіття топологій, алгоритмів маршрутизації, варіантів розміщення прикладних програм і наборів даних по вузлах мережі, способів взаємодії користувачів мережі.

Аналітична модель мережі будується у вигляді розімкнутої експонентної мережі масового обслуговування для будь-якої топології мережі, що задається аналітично або графічно. На основі аналітичної моделі вирішується завдання визначення пропускних здатностей каналу зв'язку у розподілених мережі при обмеженнях на час доставки пакетів або на вартість мережі.

Програма дозволяє розрахувати характеристики мережі, такі як:

– завантаження кожного каналу зв'язку;

– інтенсивності потоків пакетів у каналах зв'язку;

– пропускні здатності каналу зв'язку;

– час затримки пакетів при передачі по кожному каналу й у мережі в цілому;

– імовірності передачі пакетів від користувачів у мережу, між каналами й з мережі до користувачів.

Програма дозволяє варіювати отримані характеристики, такі як:

– завантаження каналу зв'язку;

– час доставки пакетів у мережі;

– пропускні здатності;

– час передачі по каналу зв'язку;

– вартість мережі для одержання необхідних результатів.

Крім того, вона дозволяє вибрати значення пропускних здатностей з дискретного ряду значень, знайдених із традиційної оптимізації безперервних значень пропускних здатностей.

Імітаційна модель мережі генерується автоматично на основі розрахунку характеристик аналітичної моделі у вигляді розімкнутої експонентної експонентної мережі масового обслуговування. Отримані характеристики аналітичної моделі використовуються як параметри імітаційної моделі мережі, що представляє собою розімкнуту експонентної мережі масового обслуговування, вузли якої відповідають каналам зв'язку, а заявки – пакетам у мережі. Число джерел заявок в імітаційній моделі дорівнює числу вузлів у мережі, при цьому інтенсивності надходження заявок у моделі визначаються як зовнішні інтенсивності пакетів від користувачів до вузлів мережі відповідно. Аналітична модель, реалізована в програмі, дозволяє розрахувати наступні параметри для імітаційної моделі.

1. Імовірність передачі пакетів від користувача *j* до каналу *k*.

2. Імовірність передачі пакетів від каналу *k* до каналу *h*.

3. Імовірність передачі пакетів від каналу *k* до користувача *j*.

4. Середній інтервал часу між вступниками пакетами від користувача *j* у мережі.

5. Середній час передач пакетів у каналах зв'язку *k*.

Крім **розрахованих параметрів** для імітаційної моделі мережі додатково необхідно **задати наступні параметри**:

а) закони розподілів інтервалів часу між вступниками пакетами від користувачів у мережі із коефіцієнтами варіації цих розподілів;

б) закони розподілів часу передачі пакетів у каналу зв’язку із коефіцієнтами варіації цих розподілів.

Відзначимо, що імітаційна модель мережі призначена для детального аналізу характеристик функціонування мережі, спроектованої в процесі

аналітичного моделювання. При цьому, у випадку відмінності реального характеру процесів надходження пакетів у мережу або передачі пакетів по каналах зв'язку від експонентного, в імітаційній моделі передбачена можливість варіювання законів розподілу часу передачі (довжин обслуговування) пакетів у кожному із каналу зв’язку, а також законів розподілу інтервалів часу між вступними в мережу пакетами. Як такі закони в роботі використовувалися наступні розподіли: експонентний, детермінований; гіпоекспоненційний різного порядку й, відповідно, з різними коефіцієнтами варіації; рівномірний; експонентний з ненульовими зсувами; гіперекспонентний; Гамма-розподіл.

Аналогічно **розроблений засіб дослідження часових характеристик каналу зв'язку** шляхом генерування імітаційної моделі каналу зв'язку у вигляді СМО типу G/G/1 з можливістю зміни завантаження каналу й варіювання законів розподілу інтервалів часу між пакетами й часу передачі пакетів по каналі. Даний засіб дозволяє одержати значення часових характеристик каналу зв'язку й оцінити погрішність аналітичних методів розрахунку характеристик каналу зв'язку.

Сформульовано **методику визначення пропускної здатності мережі на основі методології RFC6349** із неоднорідним трафіком, що містить наступні етапи.

1. Підготовка вихідних даних для програмного комплексу:

– кількість вузлів мережі і їхнє взаємне розташування;

– навантаження мережі, створювана користувачами при роботі із прикладними програмами, наборами даних і в процесі обміну повідомлень;

– обмеження на середній час доставки пакетів або на вартість мережі;

– максимальна довжина пакета;

– вибір типу каналів і завдання вартісних коефіцієнтів каналу зв’язку.

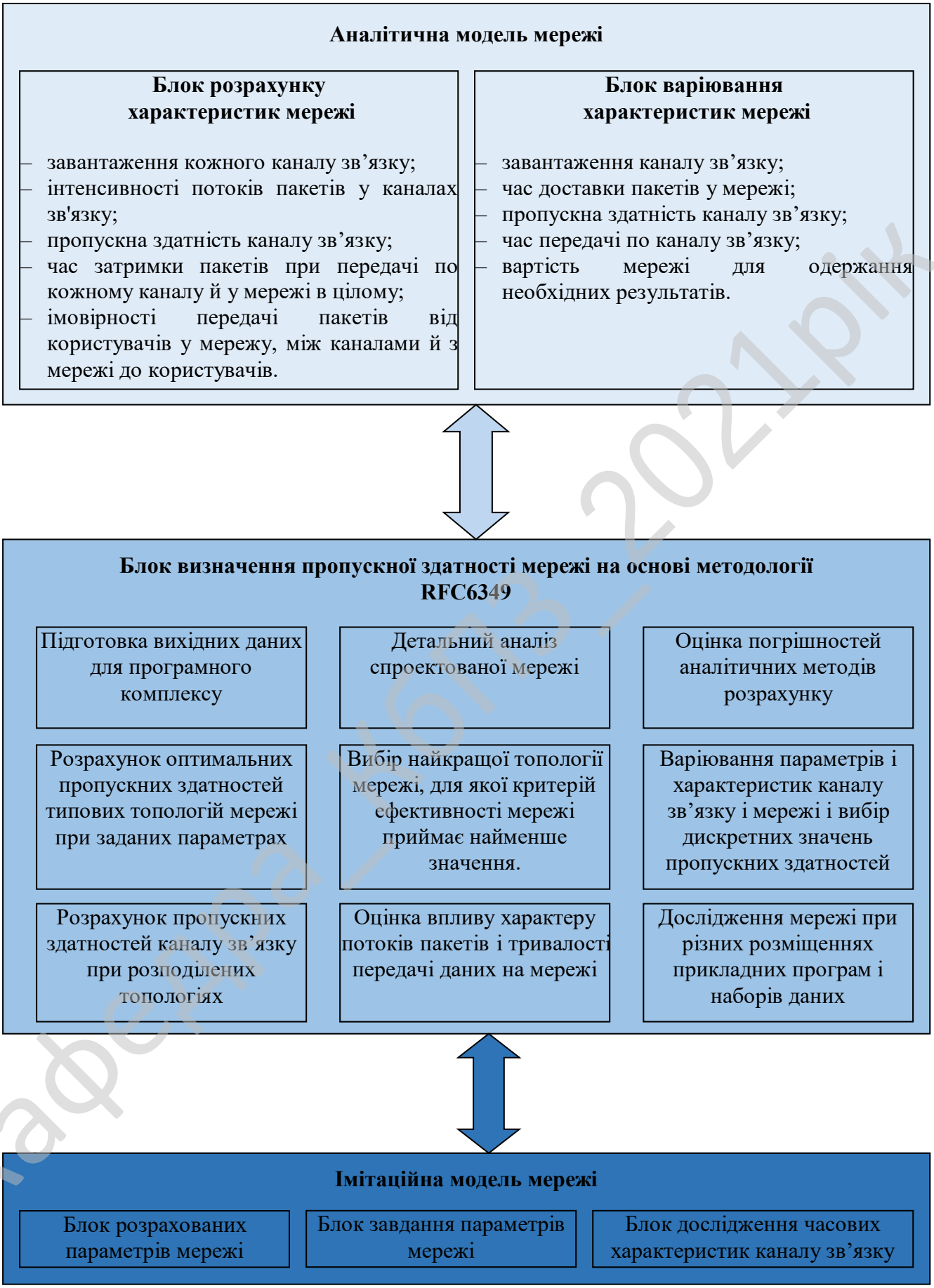

Рисунок 3.2 – Функціональна схема системи

2. Розрахунок оптимальних пропускних здатностей типових топологій мережі при заданих наступних параметрах:

– типова топологія: зірка, кільце, дерево, повнозв’язна;

– модель взаємодії користувачів мережі: RDA (Remote Data Access), DBS (DataBase Server) і AS (Application Server);

– розподіл прикладних програм і наборів даних по вузлах мережі;

– метод маршрутизації.

Після завдання цих параметрів розраховуються оптимальні значення пропускних здатностей каналу зв’язку, час передачі пакетів і завантаження каналів.

3. Розрахунок пропускних здатностей каналу зв’язку при розподілених топологіях.

4. Вибір найкращої топології мережі, для якої обрана залежно від постановки завдання як критерій ефективності характеристика мережі (середній час доставки пакетів або вартість мережі) приймає найменше значення.

5. Варіювання параметрів і характеристик каналу зв’язку і мережі і вибір дискретних значень пропускних здатностей.

6. Оцінка погрішностей аналітичних методів розрахунку характеристик каналу зв’язку.

7. Оцінка впливу характеру потоків пакетів і тривалості передачі даних на характеристики мережі:

– вплив третього моменту розподілу вхідного потоку пакетів на характеристики каналу зв’язку;

– вплив довжини пакетів на характеристики каналів зв'язку й мережі;

– вплив законів розподілів трафіку в мережах і часі передачі пакетів у каналу зв’язку на характеристики функціонування мережі.

8. Дослідження мережі при різних розміщеннях прикладних програм і наборів даних.

9. Детальний аналіз спроектованої мережі, що припускає варіювання параметрів мережі, у тому числі: довжини запитів і відповідей прикладних програм і наборів даних, імовірність передачі по основному шляху, способи маршрутизації й т.д.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

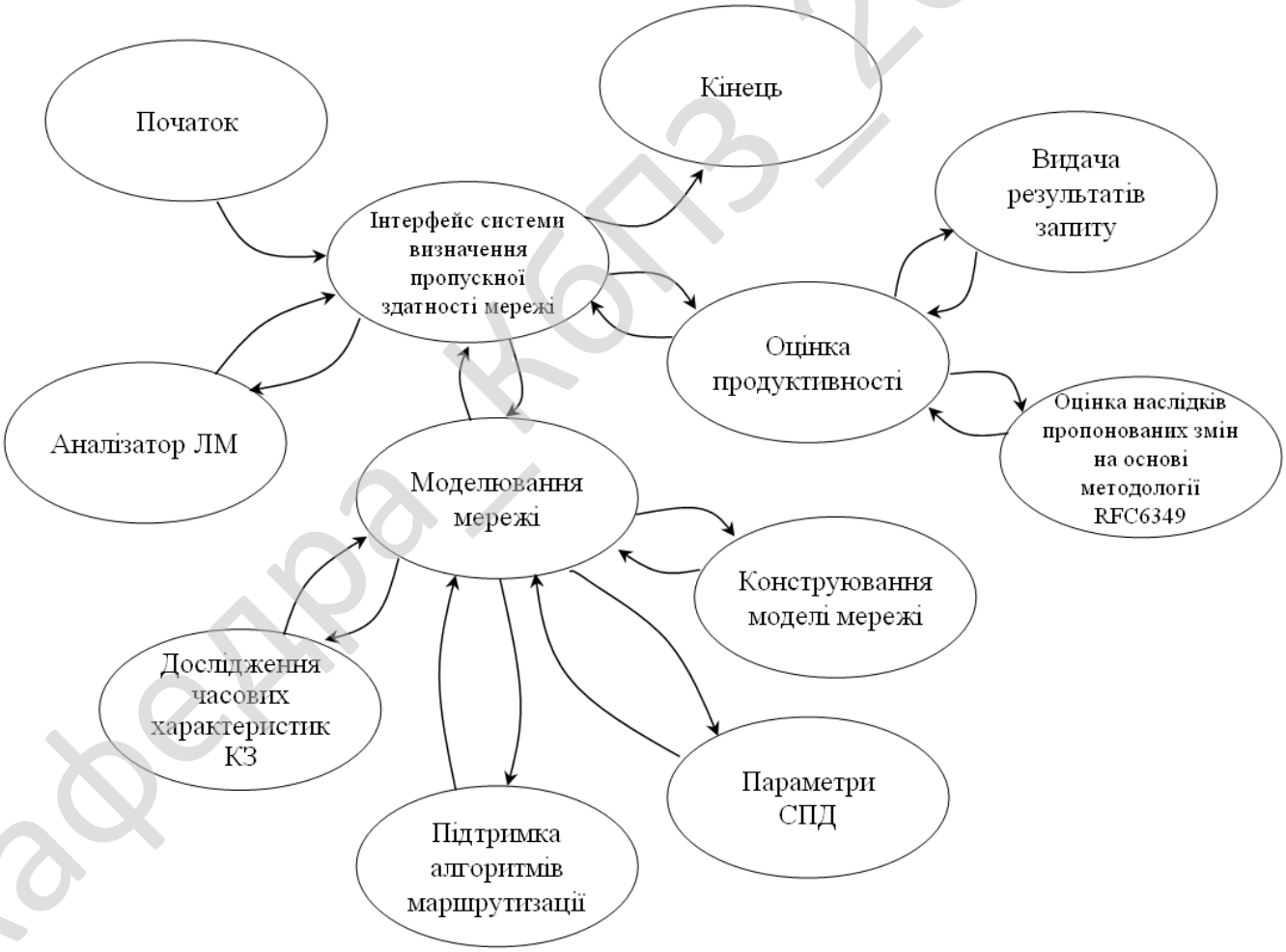

Рисунок 3.3 – Діаграма взаємодії процесів

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, поток сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку 4.1 видно, що після запуску програми відбуваються наступні дії:

– Збір інформації про пропускну здатність мережі;

– Обробка отриманої інформації на основі методології RFC6349;

– Виведення інформації про ЛМ;

– Виведення графіків вхідного TCP/IP трафіку;

– Виведення графіків вихідного TCP/IP трафіку;

– Аналіз та виведення пропускної здатності ЛМ;

– Виведення статистичних даних часу затримки пакетів;

– Виведення завантаження кожного ресурсу ЛМ;

– Виведення інтенсивності потоків пакетів

– Запит налаштування;

– Налаштування змінних характеристик ресурсів;

– Отримання та аналіз внесених змін;

– Виведення отриманих результатів.

Робота підпрограми зображено на рисунку 4.2. З рисунку 4.2 видно, що після запуску програми відбуваються наступні дії:

– Підготовка вихідних даних пропускної здатності визначеної мережі;

– Розрахунок оптимальних пропускних здатностей визначеної мережі при заданих параметрах;

– Розрахунок пропускних здатностей КЗ при розподілених топологіях;

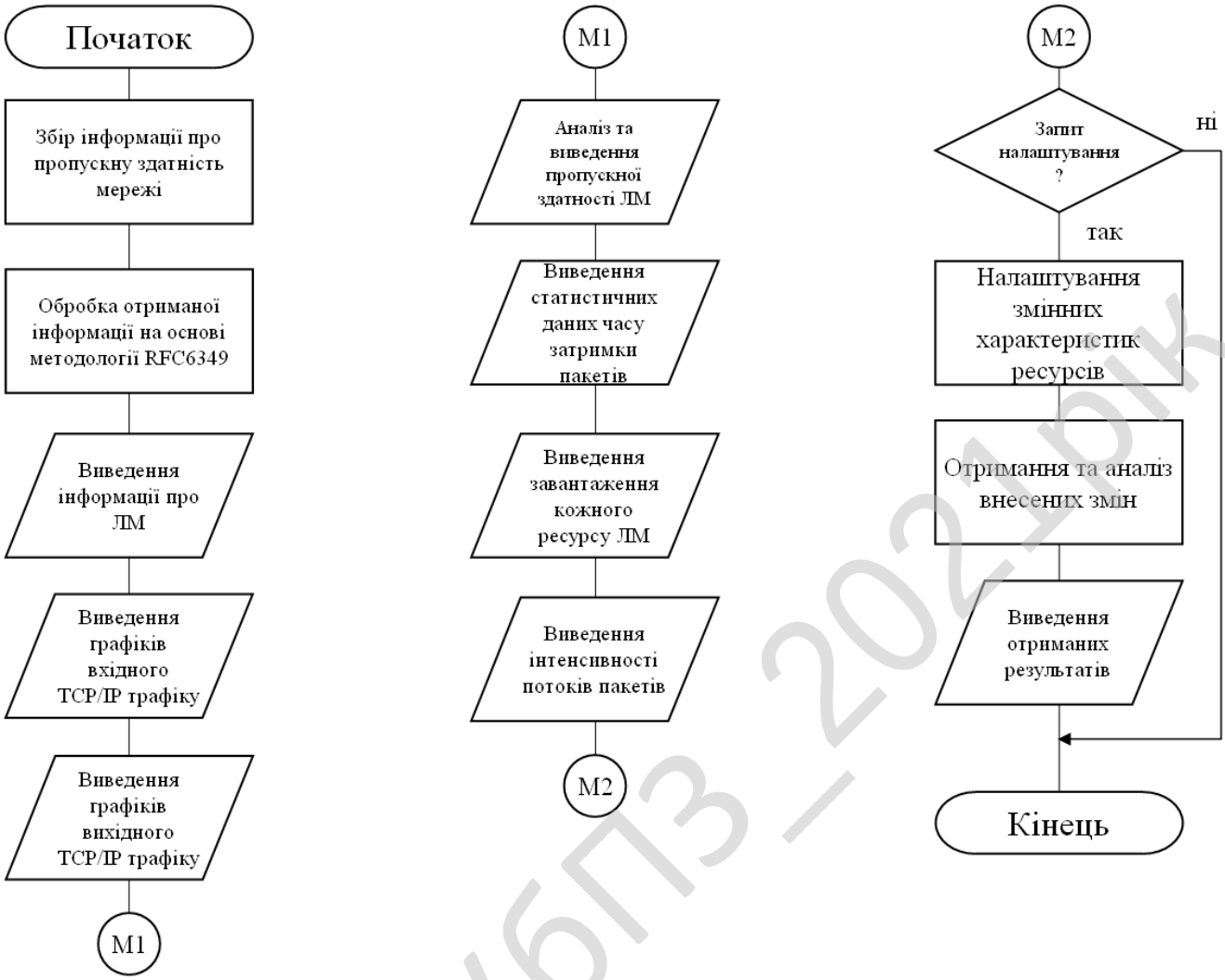

Рисунок 4.1 – Блок-схема основної програми

– Аналіз спроектованої системи;

– Вибір необхідної топології з обраними критеріями ефективності;

– Оцінка погрішностей аналітичних методів розрахунку;

– Варіювання параметрів і характеристик мережі;

– Вибір дискретних значень пропускних здатностей;

– Дослідження мережі при застосуванні різних прикладних програм і наборів даних;

– Виведення результатів на екран;

– Запит збереження результатів оцінки продуктивності у файлі формату *.dat.

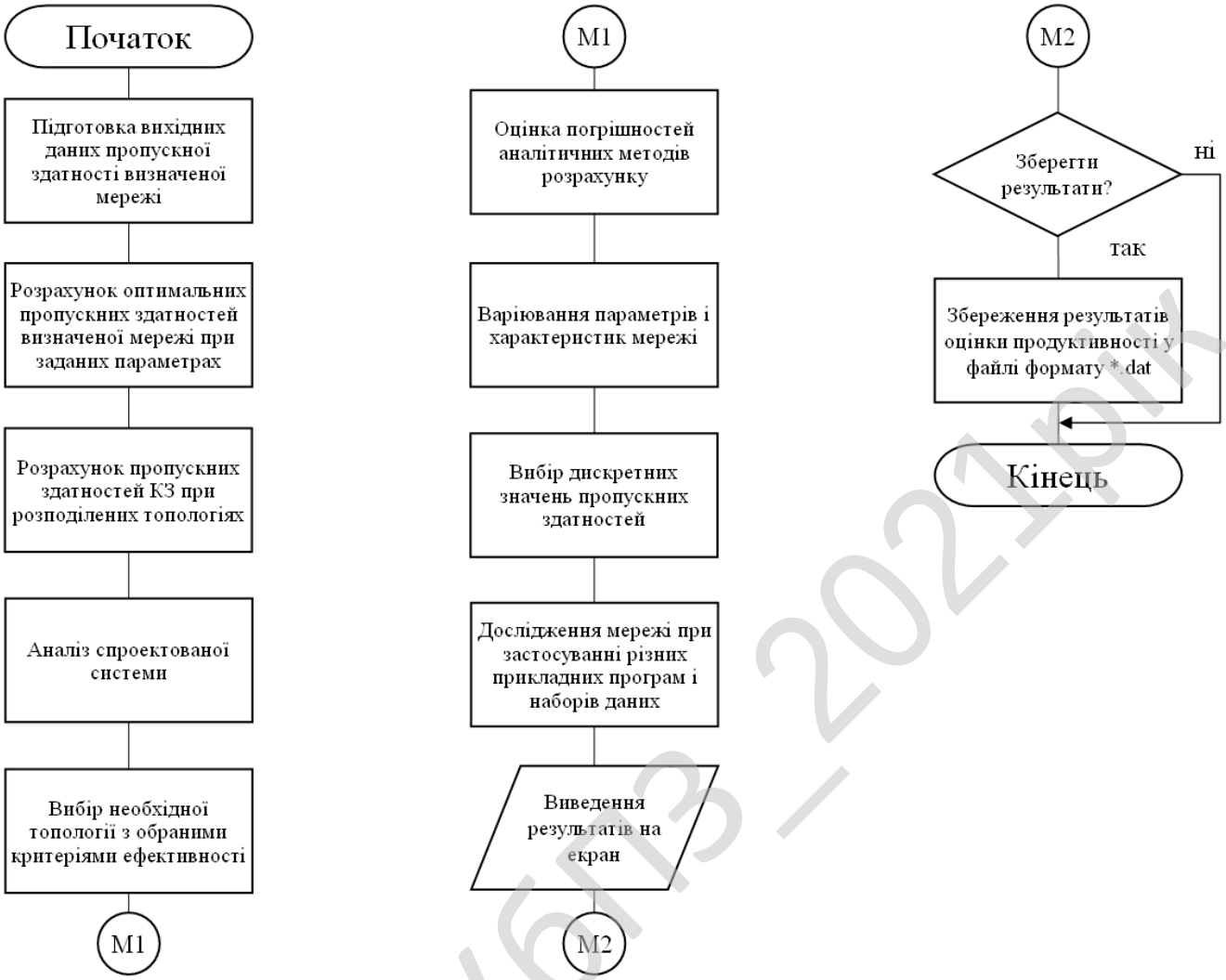

Рисунок 4.2 – Блок-схема роботи підпрограми

Розглянемо вихідний код модуля реалізації шифрування методом Виженера для підтримки реалізації методології RFC6349. Вихідний код:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Spin;
type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
```

```
    Edit1: TEdit;
    x1: TSpinEdit;
    x2: TSpinEdit;
    GroupBox2: TGroupBox;
    Memo1: TMemo;
    Memo2: TMemo;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
private
    { Declarations privées }
      function Crypter:boolean;
      function Decrypter:boolean;
      function AlgoCesar(decalage:integer;texte:string):string;
      function VigenereCrypt(password,texte:string):string;
      function VigenereDecrypt(password,texte:string):string;
public
    { Declarations publiques }
end;
var
  Form1: TForm1;

implementation

{$R *.DFM}
function Tform1.AlgoCesar(decalage:integer;texte:string):string;
var tmp:string;
    i:integer;
begin
  tmp:='';
  for i:=1 to length(texte) do
    begin
      tmp:=tmp+chr(ord(texte[i])+decalage);
    end;
  AlgoCesar:=tmp;
end;

// Фишрування
function Tform1.VigenereCrypt(password,texte:string):string;
var i,nb:integer;
    tmp:string;
```

```
begin
 tmp:='';
 nb:=1;
for i:=1 to length(texte) do
  begin
      tmp:=tmp+chr(ord(texte[i])+ord(password[nb]));
      nb:=nb+1;
      if nb>length(password) then
        begin
          nb:=1;
  end;
end;
  VigenereCrypt:=tmp;
end;

// Дешишрування
function Tform1.VigenereDecrypt(password,texte:string):string;
var i,nb:integer;
    tmp:string;
begin
  tmp:='';
  nb:=1;
  for i:=1 to length(texte) do
    begin
      tmp:=tmp+chr((ord(texte[i])+256)-ord(password[nb]));
      nb:=nb+1;
      if nb>length(password) then
        begin
          nb:=1;
        end;
    end;
  VigenereDecrypt:=tmp;
end;

 // тестове поле виведення результату
function Tform1.Crypter:boolean;
begin
memo2.text:=VigenereCrypt(AlgoCesar(x2.value,edit1.text),AlgoCesar((0-
x1.value),memo1.text));
if AlgoCesar(x1.value,VigenereDeCrypt(AlgoCesar(x2.value,
             edit1.text),memo2.text))=memo1.text then
begin
```

```
        crypter:=true;
    end
    else
    begin
        crypter:=false;
    end;
    end;

    function Tform1.Decrypter:boolean;
    begin
    memo1.text:=AlgoCesar(x1.value,VigenereDeCrypt(
                        AlgoCesar(x2.value,edit1.text),memo2.text));
    end;
    procedure TForm1.Button1Click(Sender: TObject);
    begin
      if crypter=true then
        begin
          showmessage('Crypt');
        end
        else
        begin
          showmessage('Decrypt');
        end;
    end;
    procedure TForm1.Button2Click(Sender: TObject);
    begin
    memo1.text:=AlgoCesar(x1.value,VigenereDeCrypt(
                      AlgoCesar(x2.value,edit1.text),memo2.text));
    end;
    end.
```

Розглянемо формат що використовується – JSON (JavaScript Object Notation, укр. запис об'єктів JavaScript, вимовляється джейсон) – це текстовий формат обміну даними між комп'ютерами.

JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією). Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

JSON з'явився через необхідність обміну даними з сервером у реальному часі без використання плагінів для браузерів, flash-додатків або Java-аплетів, які використовувались скрізь на початку 2000-х років. Дуглас Крокфорд був тим, хто активно просував новий на той час формат. Він з колегами хотів створити технологію, яка використовувала б можливості звичайного браузера давала б веб-розробникам можливість створювати веб-додатки із постійним двостороннім зв'язком із веб-сервером.

JSON вперше був використаний в проєкті в Communities.com для Cartoon Network, він дозволяв обмінюватись повідомленнями і одночасно маніпулювати DHTML-елементами.

Веб-сайт JSON.org було запущено 2002 року. У грудні 2005-го року Yahoo! почав переводити деякі зі своїх веб-сервісів на роботу з JSON. Google взялася до роботи з технологією у своєму веб-протоколі GData у грудні 2006-го року.

Використання

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-застосунки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси).

Формат JSON так само добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

Синтаксис

JSON будується на двох структурах:

1. Набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом.

2. Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

1. Об'єкт – це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.

2. Масив – це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

3. Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

4. Рядок – це послідовність з нуля аб о більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються зі зворотної косої риски «\». Символи представляються простим рядком. Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже сх оже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Використання JSON в AJAX

Наступний фрагмент коду JavaScript показує, як клієнт може використати XMLHttpRequest для запиту об'єктів в форматі JSON з серверу. (Серверна частина коду пропущена, вона просто повертає на запит URL рядок у JSON форматі).

Треба відзначити, що тут використання XMLHttpRequest не є кросс - браузерним (за деталями звертайтеся на сторінку XMLHttpRequest), і код зазнаватиме незначних модифікацій в різних версіях оглядачів Internet Explorer, Opera, Safari або Mozilla. Використання запиту XMLHttpRequest обмежено правилом одного джерела (same origin policy): URL, що відповідає на запит, має посилатися на той же сайт, що обслуговує сторінку, що ініціювала запит.

Оглядачі можуть також використовувати тег <iframe> для асинхронного запиту JSON-даних в кросс-браузерному варіанті, або використати просте перенаправлення <form action="url_to_cgi_script" target= "name_of_hidden_iframe">. Такий підхід був поширений до приходу популярного нині запиту XMLHttpRequest.

Динамічний тег <script> також можна використати для підвантаження JSON-даних. Ця техніка можлива, щоб обійти надто суворе правило одного джерела, але вона не є безпечною. Запит JSONRequest пропонується, як безпечніша альтернатива.

Питання безпеки

Хоча JSON призначений для передачі даних в серіалізованому вигляді, його синтаксис відповідає синтаксису JavaScript і це створює низку проблем безпеки. Часто для обробки даних, отриманих від зовнішнього джерела у форматі JSON, до них застосовується функція eval() без якої -небудь попередньої перевірки.

JavaScript eval()

Оскільки JSON представляється синтаксично правильним фрагментом коду JavaScript, природним способом розбору JSON-даних в JavaScript-програмі є використання вбудованої в JavaScript функції eval(), яка призначена для

обчислення JavaScript-виразів. При цьому підході відпадає необхідність у використанні додаткових парсерів.

Техніка використання eval() робить систему вразливою, якщо джерело JSON-даних, що використовуються, не відноситься до надійних. Такими даними може виступати шкідливий JavaScript-код для атак за допомогою ін'єкції коду. За допомогою цієї вразливості можливо здійснювати крадіжку даних, підробку автентифікації. Проте, вразливість можна усунути за рахунок використання додаткових засобів перевірки даних на коректність. Наприклад, до виконання eval() отримані від зовнішнього джерела дані можуть перевірятися за допомогою регулярних виразів. У RFC, що визначає JSON пропонується використовувати такий код для перевірки його відповідності формату JSON

```
const my_JSON_object = !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
text.replace(/"(\\.|[^"\\])*"/g, '''))) &&
eval('(' + testedData + ')');
```

Як безпечніша альтернатива eval() була запропонована нова функція parseJSON(), здатна обробляти тільки JSON-дані.

Вбудований JSON

Останні версії веб-браузерів мають вбудовану підтримку JSON і здатні його обробляти без виклику функції eval(), що призводить до описаної проблеми. Обробка JSON у такому разі зазвичай здійснюється швидше. Так у червні 2009 року вбудовану підтримку JSON мали такі браузери: Mozilla Firefox 3.5+; SeaMonkey 2; Thunderbird 3; Microsoft Internet Explorer 8; Opera 10.5; Браузери, засновані на WebKit (наприклад, Google Chrome, Apple Safari).

Принаймні дві популярні бібліотеки JavaScript використовують вбудований JSON у разі його доступності: jQuery; Dojo.

Підробка крос-доменного запиту

Непродумане використання JSON робить сайти вразливими до підробки міжсайтових запитів (CSRF або XSRF). Оскільки тег <script> допускає використання джерела, що не належить до того ж домену, що і використовуваний ресурс, це дозволяє виконувати код даних, представлених у форматі JSON, в

контексті довільної сторінки, що робить можливою компрометацію паролів або іншої конфіденційної інформації користувачів, що пройшли авторизацію на іншому сайті.

Це є проблемою тільки у разі вмісту в JSON -даних конфіденційної інформації, яка може бути компрометована третьою стороною і якщо сервер розраховує на політику одного джерела, блокуючи доступ до даних при виявленні зовнішнього запиту.

Це не є проблемою, якщо сервер визначає допустимість запиту, надаючи дані тільки у разі його коректності. HTTP cookie не можна використовувати для визначення цього. Виключне використання HTTP cookie використовується підробкою міжсайтових запитів.

Розширення, JSONP

JSONP або «JSON з підкладкою» є розширенням JSON, коли назва функції зворотного виклику вказується як вхідний аргумент.

Спочатку ідея була запропонована в блозі MacPython в 2005 році, і в наш час використовується багатьма Web 2.0 застосунками, такими, як Dojo Toolkit Applications, Google Toolkit Applications і zanox Web Services.

Подальші розширення цього протоколу були запропоновані з урахуванням введення додаткових аргументів, як, наприклад, у разі JSONPP за підтримки S3DB вебсервісів.

Оскільки JSONP використовує скрипт-теги, виклики по суті відкриті світові. З цієї причини JSONP може бути недоречними для зберігання конфіденційних даних.

Включення скриптових тегів від віддалених сайтів дозволяє їм передати будь-який контент на сайті. Якщо віддалений сайт має вразливості, які дозволяють виконати ін'єкції JavaScript, то початковий сайт також може бути ними зачеплений.

Розширення, BSON

BSON – це бінарна форма представлення простих структур даних і асоціативних масивів (які називають об'єктами або документами). Назва «BSON» заснована на визначенні JSON і неофіційно означає «Binary JSON» (бінарний JSON).

JSON Reference

Стандарт JSON не описує посилання на інші об'єкти або частини, але існує чернетка стандарту IETF для посилань на об'єкти на основі JSON.

Посилання дозволяють здійснювати трансклюзію - вставляти одні документи в інші.

JSON Reference - це JSON-об'єкт з ключем "$ref" (всі інші ключі ігноруються) і значенням стрічкового типу що містить URI, наприклад:

```
{ "$ref": "http://example.com/example.json#/foo/bar" }
```

Якщо URI містить ідентифікатор фрагмента (в прикладі вище "/foo/bar"), він інтерпретується як JSON Pointer.

Модуль dojox.json.ref в Dojo toolkit, забезпечує підтримку декількох форм JSON Reference.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 28147:2009, що є кла сичним алгоритмом симетричного шифрування на основі мережі Фейстеля (рисунок 4.3).

Даний алгоритм шифрує інформацію блоками по 64 біта (такі алгоритми називаються "блоковими").

Зміст мережі Фейстеля полягає в тому, що блок шифруемої інформації розбивається на два або більше субблоків, частина яких обробляється за певним законом, після чого результат цієї обробки накладається (операцією побітового додавання за модулем 2) на необроблювані субблоки.

Потім субблоки міняються місцями, після чого обробляються знову й т.д. певне для кожного алгоритму число раз – раундів.

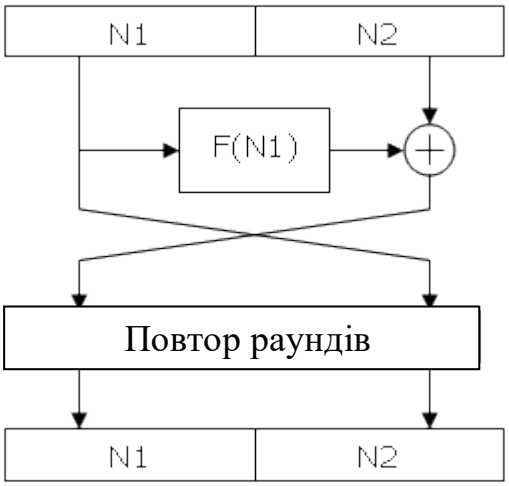

Рисунок 4.3 – Мережа Фейстеля

Основна відмінність алгоритмів симетричного шифрування друг від друга складається саме в різних функціях обробки субблоків. Дана функція часто називається "основним криптографічним перетворенням", оскільки саме вона несе основне навантаження при шифруванні інформації.

Основне перетворення алгоритму ДСТУ 28147:2009 є досить простим, що забезпечує високу швидкодію алгоритму; у ньому виконуються наступні операції (рисунок 4.4).

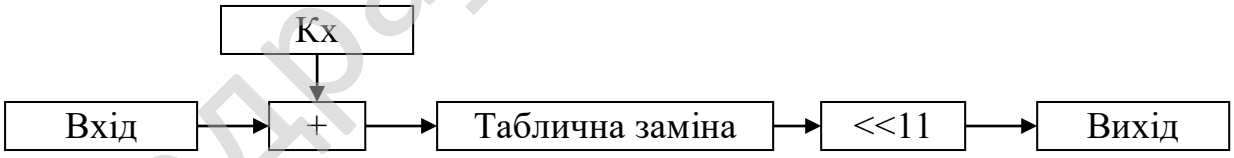

Рисунок 4.4 – Основне перетворення алгоритму ДСТ 28147:2009

1. Додавання субблоку з певним фрагменом ключа шифрування за модулем 2^{32}. Кх – це 32-бітна частина ("підключ") 256-бітного ключа шифрування, якому можна представити як конкатенацію 8 підключів: К =

$K_0K_1K_2K_3K_4K_5K_6$ K_7. Залежно від номера раунду й режиму роботи алгоритму (про їх – нижче), для даної операції вибирається один з підключів.

2. Таблична заміна. Для її виконання субблок розбивається на 8 4-бітних фрагментів, кожний з яких прогоняється через свою таблицю заміни.

Таблиця заміни містить у певній послідовності значення від 0 до 15 (тобто всі варіанти значень 4-бітні фрагменти даних); на вхід таблиці подається блок даних, числове подання якого визначає номер вихідного значення.

Наприклад, подається значення 5 на вхід наступної таблиці: "13 0 11 74 91 10 143 5 122 15 8 6". У результаті на виході виходить значення 9 (оскільки 0 заміняється на 13, 1 – на 0, 2 – на 11 і т.д.).

3. Побітове циклічне зрушення даних усередині субблока на 11 біт уліво.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Верхнього меню: Файл; Аналізатор мережі; Оцінка продуктивності; Параметри; Довідка.

– Розділу обрання групи мережних даних.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Розділу інформації про мережний інтерфейс.

– Розділу виведення результату роботи системи у графічному вигляді (іконки).

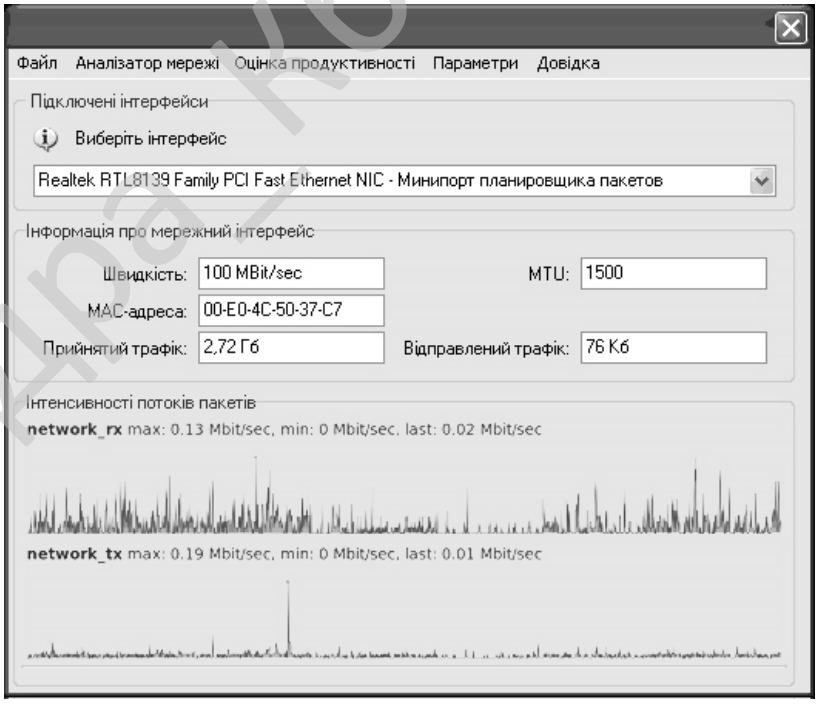

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

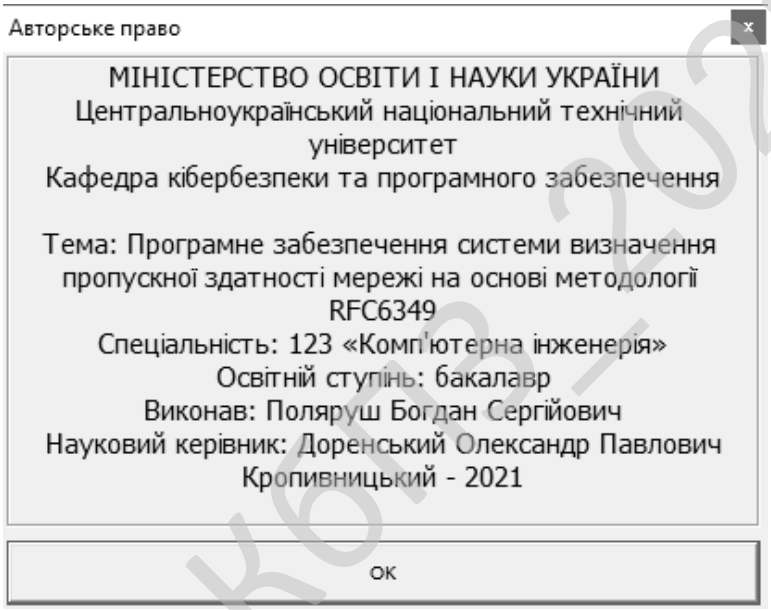

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи визначення пропускної здатності мережі на основі методології RFC6349.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем визначення пропускної здатності мережі на основі методології RFC6349.

– Досліджена система визначення пропускної здатності мережі на основі методології RFC6349.

– На основі отриманих результатів досліджень створена програмна реалізація системи визначення пропускної здатності мережі на основі методології RFC6349.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання визначення пропускної здатності мережі на основі методології RFC6349.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи визначення пропускної здатності мережі на основі методології RFC6349. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід,

зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТ 28147:2009.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб.: Питер, 2007. – 958 с.

2. Партыка С.А. Метод ускоренной коррекции SPT с использованием динамических алгоритмов [Електронний ресурс]. – Режим доступу до ресурсу: http://openarchive.nure.ua/bitstream/123456789/936/1/ASU_158_2012%20%2842-47%29.pdf

3. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

4. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

5. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

6. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

7. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. –

Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

8. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

9. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

10. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

11. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

12. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

13. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

14. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

15. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

16. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

17. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

18. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

19. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

20. Смирнов С. А. Комплекс GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

21. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

22. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

23. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

24. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

25. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

26. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та

захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

27. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

28. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

29. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

30. Смирнов С. А. Технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

31. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

32. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

33. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

34. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

35. Смирнов С. А. Разработка комплекса GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук. -практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

36. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

37. Смирнов С. А. GERT-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського

суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

38. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук. -практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

39. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

40. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

41. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук. -практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

42. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-

практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

43. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

44. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

45. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і персп ективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

46. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

47. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая - 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

48. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. - техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

49. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

50. Столлингс В. Современные компьютерные сети / Вильям Столлингс. – СПб.: Питер, 2003. – 778 с.

51. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					КБР-123.21.0039.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Поляруш Б.С.			Програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Доренський О.П.				Б	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-18-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи визначення пропускної здатності мережі на основі методології RFC6349.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи визначення пропускної здатності мережі на основі методології RFC6349;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з с умісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 3.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Доренський О.П.

Програмне забезпечення системи визначення пропускної здатності мережі на основі методології RFC6349

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл Main.pas основної програми

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Dim, Networks, StdCtrls, ComCtrls, ImgList, ShellAPI, ShlObj, IpHlpApi,
IPtraff,
  ExtCtrls, About, Buttons;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    ClickMe: TButton;
    TreeView1: TTreeView;
    ImageList1: TImageList;
    ListView1: TListView;
    Memo2: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Memo3: TMemo;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    MemoTR: TMemo;
    Timer1: TTimer;
    Label7: TLabel;
    Button2: TButton;
    Button3: TButton;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    MemoX: TMemo;
    Label8: TLabel;
    procedure ClickMeClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton4Click(Sender: TObject);

  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

function GetShellFolder(CompName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=CompName;
 if Pos(' \\' , S) <> 1 then S:=' \\' +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 Memo3.Lines.Clear;
 Screen.Cursor:=crHourGlass;
 try
  if not EnumSharedResources(Edit1.Text, Memo3.Lines)
   then raise Exception.Create(' Невірне ім"я комп"ютера' );
 finally
  Screen.Cursor:=crDefault;
 end;
end;



procedure TForm1.ClickMeClick(Sender: TObject);
var
 N: TNetworkNeighborhood;
 List: TStringList;
 i, j: Integer;
 ListViewItem: TListItem;
 WorkgroupNode, ComputerNode: TTreeNode;
begin
 Screen.Cursor:=crHourGlass;
 try
  // Ініціалізація об' єктів та сканування досліджуваної мережі з неоднорідним
трафіком
  N:=TNetworkNeighborhood.Create;
  try
   // Отримання списку всіх комп'ютерів в досліджуваній мережі з неоднорідним
трафіком
   Memo1.Lines.Clear;
   N.ListComputers(Memo1.Lines);

   // Отримання списку всіх робочих груп і комп'ютерів в досліджуваній мережі з
неоднорідним трафіком, відсортованих в алфавітному порядку
   List:=TStringList.Create;
   ListView1.Items.Clear;
   try
    N.ListNetwork(List);
    for i:=0 to List.Count - 1 do begin
     ListViewItem:=ListView1.Items.Add;
     ListViewItem.Caption:=List[i];
     ListViewItem.ImageIndex:=Integer(List.Objects[i]);
    end;
   finally
    List.Free;
   end;
```

```
   // Побудова дерева робочих груп і комп'ютерів в досліджуваній мережі з
неоднорідним трафіком
   TreeView1.Items.Clear;
   for i:=0 to N.Count - 1 do begin
    WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
    WorkgroupNode.ImageIndex:=1;
    WorkgroupNode.SelectedIndex:=1;
    for j:=0 to (N.Objects[i] as TStrings).Count - 1 do begin
     ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode, (N.Objects[i] as
TStrings).Strings[j]);
     ComputerNode.ImageIndex:=0;
    end;
   end;

  // Отримання IP адрес комп'ютерів
  GetIPAddresses(N, Memo2.Lines);

  finally
   N.Free;
  end;
  TreeView1.FullExpand;

 finally
  Screen.Cursor:=crDefault;
 end;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:=GetComputerName;

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end;

end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin

   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;

end;

procedure TForm1.DOIpStuff;
begin

 Get_TCPTable( MemoX.Lines );
 Get_UDPTable( MemoTR.Lines );

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Form1.Close;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
```

```
begin
Form1.Close;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
Form2.Show;
end;


end.
```

Файл NetConst.pas – ініціалізація констант

```
unit NetConst;

interface

resourcestring
  SShellLinkReadError = ' Помилка  читання' ;
  SShellLinkWriteError = ' Помилка запису' ;
  SShellLinkLoadError = ' Не можу завантажити %s' ;
  SShellLinkSaveError = ' Не можу зберегти %s' ;
  SShellLinkCreateError = ' Не можу ініціалізувати інтерфейс' ;
  SDynArrayIndexError = ' У масиві %s немає елемента з таким індексом (%d)' ;
  SDynArrayCountError = ' Перевищена довжина масиву (%d)' ;
  SSharedMemoryError = ' Не можу створити файл' ;
  SCannotInitTimer = ' Не можу ініціалізувати таймер' ;
  SPrinterIndexError = ' Принтер не доступний (%d)' ;
  SIndicesOutOfRange = ' Недопустимий індекс матриці [%d: %d]' ;
  SRowIndexOutOfRange = ' Недопустиме значення рядка матриці (%d)' ;
  SColIndexOutOfRange = ' Недопустиме значення стовбчика матриці (%d)' ;
  SNoAdminRights = ' У Вас немає прав адміністратора' ;

  SFileError = ' Помилка %s файл %s%s' ;
  SFileReading = ' читання' ;
  SFileWriting = ' запис' ;
  SFileError002 = '  - файл не знайдено' ;
  SFileError003 = '  - шлях не знайдено' ;
  SFileError004 = '  - не можу відкрити файл' ;
  SFileError005 = '  - немає доступу' ;
  SFileError014 = '  - не достатньо пам"яті' ;
  SFileError015 = '  - не можу знайти драйвер' ;
  SFileError017 = '  - не можу перемістити файл' ;
  SFileError019 = '  - носій захищений від запису' ;
  SFileError020 = '  - не можу знайти пристрій' ;
  SFileError021 = '  - пристрій не відкривається для читання' ;
  SFileError022 = '  - пристрій не може розпізнати команду' ;
  SFileError025 = '  - вказана область не знайдена' ;
  SFileError026 = '  - пристрій недоступний' ;
  SFileError027 = '  - сектор не знайдено' ;
  SFileError029 = '  - помилка запису на пристрій' ;
  SFileError030 = '  - помилка читання з пристрою' ;
  SFileError032 = '  - файл використовується іншою програмою' ;
  SFileError036 = '  - занадто багато відкритих файлів' ;
  SFileError038 = '  - досягнутий кінець файлу' ;
  SFileError039 = '  - диск переповнений' ;
  SFileError050 = '  - запит не підтримується' ;
  SFileError051 = '  - віддалений комп'ютер недоступний' ;
  SFileError052 = '  - у досліджуваній мережі з неоднорідним трафіком знайдені
ідентичні імена' ;
  SFileError053 = '  - мережний шлях не знайдено' ;
  SFileError054 = '  - досліджувана мережа зайнята' ;
  SFileError055 = '  - ресурс досліджуваної мережі з неоднорідним трафіком або
пристрій недоступний' ;
  SFileError057 = '  - апаратна помилка в мережному адаптері' ;
  SFileError058 = '  - сервер не в змозі виконувати дану дію' ;
  SFileError059 = '  - помилка у досліджуваній мережі з неоднорідним трафіком' ;
  SFileError064 = '  - недоступне мережне ім"я' ;
  SFileError065 = '  - немає доступу до досліджуваної мережі з неоднорідним
трафіком' ;
  SFileError066 = '  - невірно вказатий тип мережного ресурсу' ;
  SFileError067 = '  - не знайдене вказане мережне ім"я' ;
  SFileError070 = '  - відключейний сервер досліджуваної мережі з неоднорідним
трафіком' ;
  SFileError082 = '  - не можу створити файл чи каталог' ;
  SFileError112 = '  - не достатньо вільного місця на диску' ;
  SFileError123 = '  - в імені файлу вказано недопустимий символ' ;
  SFileError161 = '  - неправильно вказано шдях' ;
```

```
  SFileError183 = ' - файл не існує' ;

  SCannotSetSize = ' Не можу змінити розмір файлу' ;

  SUnableToCompress = ' Не можу заархівувати дані' ;
  SUnableToDecompress = ' Не можу розархівувати дані' ;

  SCannotFindNetwork = ' Не можу знайти досліджувану мережу' ;

implementation

end.
```

Файл Networks.pas – робота з досліджуваною мережею

```
unit Networks;

interface

uses Windows, ShellAPI, ShlObj, ActiveX, ComObj, Dim, Classes, SysUtils,
WinSock;

type
  ComputerFound = class (Exception);
  ECannotFindNetwork = class (Exception);

  TStringObject = class (TObject)
  private
    FValue: TString;
    FTag: Integer;
    FData: Pointer;
    FRefObj: TObject;
    procedure SetValue(const Value: TString);
    procedure SetData(const Value: Pointer);
    procedure SetRefObj(const Value: TObject);
    procedure SetTag(const Value: Integer);
  public
    property Value: TString read FValue write SetValue;
    property RefObj: TObject read FRefObj write SetRefObj;
    property Tag: Integer read FTag write SetTag;
    property Data: Pointer read FData write SetData;
  end;

  TStringObjectArray = class (TDynamicArray)
  private
    function GetObject(Index: Integer): TStringObject;
    function GetData(Index: Integer): Pointer;
    function GetRefObj(Index: Integer): TObject;
    function GetTag(Index: Integer): Integer;
    function GetValue(Index: Integer): TString;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetRefObj(Index: Integer; const Value: TObject);
    procedure SetTag(Index: Integer; const Value: Integer);
    procedure SetValue(Index: Integer; const Value: TString);

    function FreeObject(Index: Integer; var Obj: TStringObject): Integer;

    procedure FreeItem(Index: Integer);
    procedure CreateItem(Index: Integer);

  protected
    procedure SetCount(const NewCount: Cardinal); override;
  public
    function Add: Integer; override;
    procedure Insert(Index: Integer); override;
    procedure Delete(Index: Integer); override;
    function AddItem(const Item): Integer; override;
    procedure InsertItem(Index: Integer; const Item); override;
    procedure DeleteItem(Index: Integer; out Item); override;
    property Value[Index: Integer]: TString read GetValue write SetValue;
default;
    property RefObj[Index: Integer]: TObject read GetRefObj write SetRefObj;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;
    property Data[Index: Integer]: Pointer read GetData write SetData;
    constructor Create;
    destructor Destroy; override;
  end;

  TStringObjectList = class (TStrings)
  private
```

```
    FArray: TStringObjectArray;
    function GetData(Index: Integer): Pointer;
    function GetTag(Index: Integer): Integer;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetTag(Index: Integer; const Value: Integer);
  protected
    function Get(Index: Integer): string; override;
    function GetCount: Integer; override;
    function GetObject(Index: Integer): TObject; override;
    procedure Put(Index: Integer; const S: string); override;
    procedure PutObject(Index: Integer; AObject: TObject); override;

  public
    property Data[Index: Integer]: Pointer read GetData write SetData;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Exchange(Index1, Index2: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;


    constructor Create;
    destructor Destroy; override;
  end;

  {TNetworkWorkgroup - клас, який створює список всіх комп'ютерів в робочій
групі. Цей клас є нащадком класу TStrings і повністю сумісний з іншим нащадками
цього класу.  Об' єкти цього класу записуються у властивості об' єктів класу
TNetworkNeiborhood. Клас TNetworkNeiborhood містить об' єкти і властивості
робочих груп. }

  TNetworkWorkgroup = class (TStringObjectList);


  {TNetworkNeiborhood - клас, який створює список всіх робочих груп в
копоратвній мережі з неоднорідним трафіком}
  TNetworkNeighborhood = class (TStringObjectList)
  private
    function CreatePIDL(Size: Integer): PItemIDList;
    procedure DisposePIDL(ID: PItemIDList);
    function NextPIDL(IDList: PItemIDList): PItemIDList;
    function GetPIDLSize(IDList: PItemIDList): Integer;
    function CopyPIDL(IDList: PItemIDList): PItemIDList;
    procedure StripLastID(IDList: PItemIDList);
    function GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
    class function GetDisplayName(ShellFolder: IShellFolder; PIDL: PItemIDList):
TString;
    function OriginFolder:  IShellFolder;
    function OriginFolderNT: IShellFolder;
    class function EnumObjects(ShellFolder: IShellFolder): IEnumIDList;
    class procedure ParseFolder(Folder: IShellFolder; Items: TStringObjectList;
StorePIDLs: Boolean = False);
    class procedure ParseFolderEx(Folder: IShellFolder; Items: TStrings);

    function FreeRefObj(Index: Integer; var Obj: TStringObject): Integer;
    function GetWorkgroup(Name: TString): TNetworkWorkgroup;

  public
    { Процедура Оновлення шукає всі доступні робочі групи в досліджуваній мережі
з неоднорідним трафіком }

    procedure Refresh;

    { містить списки всіх комп'ютерів в  досліджуваній мережі з неоднорідним
трафіком}

    property Workgroup[Name: TString]: TNetworkWorkgroup read GetWorkgroup;
```

```
   { Функція FindComputer шукає комп'ютер зі вказаним ім'ям і повертає ім'я
робочої групи де його знайдено, або порожню строку
     якщо комп'ютера з таким іменем у досліджуваній мережі з неоднорідним
трафіком немає }

    function FindComputer(Name: TString): TString;

    { Процедура ListComputers копіює список всіх комп'ютерів в досліджуваній
мережі з неоднорідним трафіком
      у об' єкті TStrings}
    procedure ListComputers(Strings: TStrings);

    { Процедура ListNetwork копіює відсортований в алфавітному порядку список
всіх робочих груп і комп'ютерів в досліджуваній мережі з неоднорідним трафіком.
     Робочі групи мають ' TObject(1)'  у відповідному елементі властивості об'
єктів, а комп'ютери - ' nil' }

    procedure ListNetwork(Strings: TStrings);

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;
    constructor Create;
  end;

{ Функція GetIPAddress отримує IP адресу комп'ютера або сервера.
  Параметр NetworkName конкретизує ім'я комп'ютера або  сервера.
  Ця функція повертає адреси IP у форматі XXX.XXX.XXX.XXX у разі успішного
виконання, ' Error'  - коли неможливо ініціалізувати, ' Unknown'  – коли
параметр NetworkName посилається на неіснуючий комп'ютер або на комп'ютер  без
встановленого протоколу TCP/IP }

  function GetIPAddress(NetworkName: TString): TString;


{ GetIPAddresses отримує IP адреси всіх доступних комп'ютерів досліджуваної
мережі з неоднорідним трафіком }

procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);



{ Функція EnumSharedResources перераховує загальні ресурси досліджуваної мережі
з неоднорідним трафіком. Параметр ComputerName конкретизує
  ім'я комп'ютера. }

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;


implementation

uses NetConst;

{ TStringObject }

procedure TStringObject.SetData(const Value: Pointer);
begin
  FData := Value;
end;

procedure TStringObject.SetRefObj(const Value: TObject);
begin
  FRefObj := Value;
end;

procedure TStringObject.SetTag(const Value: Integer);
begin
```

```
  FTag := Value;
end;

procedure TStringObject.SetValue(const Value: TString);
begin
  FValue := Value;
end;

{ TStringObjectArray }

function TStringObjectArray.Add: Integer;
begin
 Result:=inherited Add;
 CreateItem(Result);
end;

function TStringObjectArray.AddItem(const Item): Integer;
begin
 Result:=Add;
end;

constructor TStringObjectArray.Create;
begin
 inherited Create(0, SizeOf(TStringObject));
end;

procedure TStringObjectArray.CreateItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 P^:=TStringObject.Create;
end;

procedure TStringObjectArray.Delete(Index: Integer);
begin
 FreeItem(Index);
 inherited;
end;

procedure TStringObjectArray.DeleteItem(Index: Integer; out Item);
begin
 Delete(Index);
end;

destructor TStringObjectArray.Destroy;
begin
 ForEach(Integer(Self), @TStringObjectArray.FreeObject);
 inherited;
end;

procedure TStringObjectArray.FreeItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 FreeAndNil(P^);
end;

function TStringObjectArray.FreeObject(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj);
 Result:=0;
end;

function TStringObjectArray.GetData(Index: Integer): Pointer;
begin
 Result:=GetObject(Index).Data;
```

```
end;

function TStringObjectArray.GetObject(Index: Integer): TStringObject;
begin
 GetItem(Index, Result);
end;

function TStringObjectArray.GetRefObj(Index: Integer): TObject;
begin
 Result:=GetObject(Index).RefObj;
end;

function TStringObjectArray.GetTag(Index: Integer): Integer;
begin
 Result:=GetObject(Index).Tag;
end;

function TStringObjectArray.GetValue(Index: Integer): TString;
begin
 Result:=GetObject(Index).Value;
end;

procedure TStringObjectArray.Insert(Index: Integer);
begin
 inherited;
 CreateItem(Index);
end;

procedure TStringObjectArray.InsertItem(Index: Integer; const Item);
begin
 Insert(Index);
end;

procedure TStringObjectArray.SetCount(const NewCount: Cardinal);
var
 i, OldCount: Integer;
begin
 OldCount:=Count;
 if NewCount > Count then begin
  inherited SetCount(NewCount);
  for i:=OldCount to NewCount - 1 do CreateItem(i);
 end else if NewCount < Count then begin
  for i:=NewCount to OldCount - 1 do FreeItem(i);
  inherited SetCount(NewCount);
 end;
end;

procedure TStringObjectArray.SetData(Index: Integer; const Value: Pointer);
begin
 GetObject(Index).Data:=Value;
end;

procedure TStringObjectArray.SetRefObj(Index: Integer;
  const Value: TObject);
begin
 GetObject(Index).RefObj:=Value;
end;

procedure TStringObjectArray.SetTag(Index: Integer; const Value: Integer);
begin
 GetObject(Index).Tag:=Value;
end;

procedure TStringObjectArray.SetValue(Index: Integer; const Value: TString);
begin
 GetObject(Index).Value:=Value;
end;

{ TStringObjectList }
```

```
function TStringObjectList.Add(const S: string): Integer;
begin
 Result:=FArray.Add;
 FArray.Value[Result]:=S;
end;

procedure TStringObjectList.Clear;
begin
 FArray.Count:=0;
end;

constructor TStringObjectList.Create;
begin
 inherited Create;
 FArray:=TStringObjectArray.Create;
end;

procedure TStringObjectList.Delete(Index: Integer);
begin
 FArray.Delete(Index);
end;

destructor TStringObjectList.Destroy;
begin
 FArray.Free;
 inherited;
end;

procedure TStringObjectList.Exchange(Index1, Index2: Integer);
begin
 FArray.Swap(Index1, Index2);
end;

function TStringObjectList.Get(Index: Integer): string;
begin
 Result:=FArray.Value[Index];
end;

function TStringObjectList.GetCount: Integer;
begin
 Result:=FArray.Count;
end;

function TStringObjectList.GetData(Index: Integer): Pointer;
begin
 Result:=FArray.Data[Index];
end;

function TStringObjectList.GetObject(Index: Integer): TObject;
begin
 Result:=FArray.RefObj[Index];
end;

function TStringObjectList.GetTag(Index: Integer): Integer;
begin
 Result:=FArray.Tag[Index];
end;

procedure TStringObjectList.Insert(Index: Integer; const S: string);
begin
 FArray.Insert(Index);
 FArray.Value[Index]:=S;
end;

procedure TStringObjectList.Put(Index: Integer; const S: string);
begin
 FArray.Value[Index]:=S;
end;
```

```
procedure TStringObjectList.PutObject(Index: Integer; AObject: TObject);
begin
 FArray.RefObj[Index]:=AObject;
end;

procedure TStringObjectList.SetData(Index: Integer; const Value: Pointer);
begin
 FArray.Data[Index]:=Value;
end;

procedure TStringObjectList.SetTag(Index: Integer; const Value: Integer);
begin
 FArray.Tag[Index]:=Value;
end;

{ TNetworkNeighborhood }

function TNetworkNeighborhood.Add(const S: string): Integer;
begin
 Result:=inherited Add(S);
 Objects[Result]:=TNetworkWorkgroup.Create;
end;

procedure TNetworkNeighborhood.Clear;
begin
 FArray.ForEach(Integer(Self), @TNetworkNeighborhood.FreeRefObj);
 inherited;
end;

function TNetworkNeighborhood.CopyPIDL(IDList: PItemIDList): PItemIDList;
var
 Size: Integer;
begin
 Size := GetPIDLSize(IDList);
 Result := CreatePIDL(Size);
 if Assigned(Result) then CopyMemory(Result, IDList, Size);
end;

constructor TNetworkNeighborhood.Create;
begin
 inherited Create;
 Refresh;
end;

function TNetworkNeighborhood.CreatePIDL(Size: Integer): PItemIDList;
var
 Malloc: IMalloc;
 HR: HResult;
begin
 Result := nil;
 HR := SHGetMalloc(Malloc);
 if Failed(HR) then  Exit;
 try
  Result := Malloc.Alloc(Size);
  if Assigned(Result) then FillChar(Result^, Size, 0);
 finally
 end;
end;

procedure TNetworkNeighborhood.Delete(Index: Integer);
begin
end;

procedure TNetworkNeighborhood.DisposePIDL(ID: PItemIDList);
var
 Malloc: IMalloc;
begin
 if ID = nil then Exit;
 OLECheck(SHGetMalloc(Malloc));
```

```
 Malloc.Free(ID);
end;

class function TNetworkNeighborhood.EnumObjects(
  ShellFolder: IShellFolder): IEnumIDList;
const
 Flags = SHCONTF_FOLDERS or SHCONTF_NONFOLDERS or SHCONTF_INCLUDEHIDDEN;
begin
 ShellFolder.EnumObjects(0, Flags, Result);
end;

function TNetworkNeighborhood.FindComputer(Name: TString): TString;
var
 i, j: Integer;
 List: TNetworkWorkgroup;
 S: TString;
begin
 Result:=' ' ;
 try
  for i:=0 to Count - 1 do begin
   List:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to List.Count - 1 do begin
    S:=List[j];
    CleanUp(S);
    if EqualText(Name, S) then begin
     Result:=Strings[i];
     raise ComputerFound.Create(' ' );
    end;
   end;
  end;
 except
  if not (ExceptObject is ComputerFound) then raise;
 end;
end;

function TNetworkNeighborhood.FreeRefObj(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj.FRefObj);
 Result:=0;
end;

class function TNetworkNeighborhood.GetDisplayName(ShellFolder: IShellFolder;
  PIDL: PItemIDList): TString;
var
 StrRet: TStrRet;
 P: PChar;
begin
 Result := ' ';
 ShellFolder.GetDisplayNameOf(PIDL, SHGDN_NORMAL, StrRet);
 case StrRet.uType of
  STRRET_CSTR: SetString(Result, StrRet.cStr, lStrLen(StrRet.cStr));
  STRRET_OFFSET: begin
   P := @PIDL.mkid.abID[StrRet.uOffset - SizeOf(PIDL.mkid.cb)];
   SetString(Result, P, PIDL.mkid.cb - StrRet.uOffset);
  end;
  STRRET_WSTR: Result := StrRet.pOleStr;
 end;
 CleanUp(Result, True);
end;

function TNetworkNeighborhood.GetPIDLSize(IDList: PItemIDList): Integer;
begin
 Result := 0;
 if Assigned(IDList) then begin
  Result := SizeOf(IDList^.mkid.cb);
  while IDList^.mkid.cb <> 0 do begin
   Result := Result + IDList^.mkid.cb;
   IDList := NextPIDL(IDList);
```

```
  end;
 end;
end;

function TNetworkNeighborhood.GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
var
 Temp: PItemIDList;
begin
 Temp := CopyPIDL(PIDL);
 if Assigned(Temp) then StripLastID(Temp);
 if Temp.mkid.cb <> 0 then Result:=Temp else Result:=nil;
end;

function TNetworkNeighborhood.GetWorkgroup(Name: TString): TNetworkWorkgroup;
var
 Index: Integer;
begin
 Index:=IndexOf(Name);
 if Index<>-1 then Result:=Objects[Index] as TNetworkWorkgroup else Result:=nil;
end;

procedure TNetworkNeighborhood.Insert(Index: Integer; const S: string);
begin
end;

procedure TNetworkNeighborhood.ListComputers(Strings: TStrings);
var
 i, j: integer;
 L: TNetworkWorkgroup;
 S: TString;
begin
 Strings.BeginUpdate;
 try
  Strings.Clear;
  for i:=0 to Count - 1 do begin
   L:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to L.Count - 1 do begin
    S:=L[j];
    CleanUp(S);
    Strings.Add(S);
   end;
  end;
 finally
  Strings.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.ListNetwork(Strings: TStrings);
var
 List: TStringList;
 i: Integer;
begin
 List:=TStringList.Create;
 try
  List.AddStrings(Self);
  for i:=0 to List.Count - 1 do List.Objects[i]:=TObject(1);
  for i:=0 to Count - 1 do begin
   List.AddStrings(Objects[i] as TStrings);
  end;
  for i:=Count to List.Count - 1 do List.Objects[i]:=nil;
  List.Sort;
  Strings.Assign(List);
 finally
  List.Free;
 end;
end;

function TNetworkNeighborhood.NextPIDL(IDList: PItemIDList): PItemIDList;
begin
```

```
 Result := IDList;
 Inc(PChar(Result), IDList^.mkid.cb);
end;

function TNetworkNeighborhood.OriginFolder: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString;
 P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 P:=StringToOleStr(S);
 Flags:=0;
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 try
   Network:=GetPrevPIDL(Workgroup);
  try
   Desktop.BindToObject(Network, nil, IShellFolder, Pointer(Result));
  finally
   DisposePIDL(Network);
  end;
 finally
  DisposePIDL(Workgroup);
 end;
end;

function TNetworkNeighborhood.OriginFolderNT: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString; W: WideString; P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
 NetShell: IShellFolder;
 Enum: IEnumIDList;
 ID: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 W:=S; P:=PWideChar(W);
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 Network:=GetPrevPIDL(Workgroup);
 Desktop.BindToObject(Network, nil, IShellFolder, NetShell);
 Enum:=EnumObjects(NetShell);
 Enum.Next(1, ID, Flags);
 NetShell.BindToObject(ID, nil, IShellFolder, Pointer(Result));
 DisposePIDL(Network);
 DisposePIDL(Workgroup);
end;

class procedure TNetworkNeighborhood.ParseFolder(Folder: IShellFolder;
  Items: TStringObjectList; StorePIDLs: Boolean);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
 Index: Integer;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
```

```
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Index:=Items.Add(S);
   if StorePIDLs then Items.Data[Index]:=ID;
  end;
 finally
  Items.EndUpdate;
 end;
end;

class procedure TNetworkNeighborhood.ParseFolderEx(Folder: IShellFolder;
  Items: TStrings);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Items.Add(S);
  end;
 finally
  Items.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.Refresh;
var
 Network: IShellFolder;
 Workgroup: IShellFolder;
 i: Integer;
begin
 try
  if WinNT and (not Win2K) then Network:=OriginFolderNT else
   Network:=OriginFolder;
  ParseFolder(Network, Self, True);
  for i:=0 to Count - 1 do begin
   Network.BindToObject(PItemIDList(Data[i]), nil, IShellFolder, Workgroup);
   ParseFolder(Workgroup, Objects[i] as TStringObjectList, False);
   Workgroup:=nil;
  end;
 except
  raise ECannotFindNetwork.Create(SCannotFindNetwork);
 end;
end;

procedure TNetworkNeighborhood.StripLastID(IDList: PItemIDList);
var
 MarkerID: PItemIDList;
begin
 MarkerID := IDList;
 if Assigned(IDList) then begin
  while IDList.mkid.cb <> 0 do begin
   MarkerID := IDList;
   IDList := NextPIDL(IDList);
  end;
  MarkerID.mkid.cb := 0;
 end;
end;


procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);
var
 Error: DWORD;
```

```
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
 i: Integer;
 TmpList: TStringList;
 S: TString;
begin
{ List.BeginUpdate;
 try}
  List.Clear;
  Error:=WSAStartup(MakeWord(1, 1), Data);
  if Error = 0 then begin
   TmpList:=TStringList.Create;
   try
    Network.ListComputers(TmpList);
    for i:=0 to TmpList.Count - 1 do begin
     HostEntry:=gethostbyname(PChar(TmpList[i]));
     Error:=GetLastError;
     if Error <> 0 then S:=‘ Unknown’  else begin
      Address:=PInAddr(HostEntry^.h_addr_list^)^;
      S:=inet_ntoa(Address);
     end;
     List.Add(Format(‘ %s [%s]’ , [TmpList[i], S]));
    end;
   finally
    TmpList.Free;
   end;
  end else begin
   List.Add(‘ Error’ );
  end;
{ finally
  List.EndUpdate;
 end;}
end;


function GetShellFolder(ComputerName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=ComputerName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;
var
 ShellFolder: IShellFolder;
begin
 ShellFolder:=GetShellFolder(ComputerName);
 Result:=Assigned(ShellFolder);
 if Result then TNetworkNeighborhood.ParseFolderEx(ShellFolder, List);
end;



function GetIPAddress(NetworkName: TString): TString;
var
 Error: DWORD;
```

```
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
begin
 Error:=WSAStartup(MakeWord(1, 1), Data);
 if Error = 0 then begin
  HostEntry:=gethostbyname(PChar(NetworkName));
  Error:=GetLastError();
  if Error = 0 then begin
   Address:=PInAddr(HostEntry^.h_addr_list^)^;
   Result:=inet_ntoa(Address);
  end else begin
   Result:=‘ Unknown’ ;
  end;
 end else begin
  Result:=‘ Error’ ;
 end;
 WSACleanup();
end;

end.
```

Файл IPtraff.pas – відслідковування та контроль трафіку

```
unit IPtraff;

interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.  0.  0.  0' ;


type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const

  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Пінгування   }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { Протокол File Transfer Protocol -
дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { Протокол File Transfer Protocol -
управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Протокол Simple Mail Transfer
Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Протокол Time Protocol }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { WHO IS сервіс  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP клієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { Стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { SUN віддалений виклик функцій }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Локальний сервіce             }
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Netw. Management
Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
    );
```

```
  const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL',' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION',' IP_HARDWARE_ERROR' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST',' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM',' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION',' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


  ARPEntryType  : array[1..4] of string = ( ' Інший' , ' Несправний' ,
    ' Динамічний' , ' Статичний'
    );
  TCPConnState  :
    array[1..12] of string =
 ( ' CLOSED' , ' READ' , ' SYN_SENT' ,
    ' SYN_RCVD' , ' ESTABLISHED' , ' FIN_WAIT1' ,
    ' FIN_WAIT2' , ' WAIT' , ' CLOSING' ,
    ' LAST_ACK' , ' WAIT' , ' DELETE_TCB'    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , ' MIL-STD-1778' ,
    ' Van Jacobson' , ' Other'  );

  IPForwTypes   : array[1..4] of string =
    ( ' other' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' OTHER' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
```

```
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ;
    LeaseExpires: LongInt;
  end ;

  TAdaptorRows = array of TAdaptorInfo ;



function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// утіліти перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;


{ перетворення точена у рядок, потім рядок знищується }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
```

```
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;


{ пертворення MAC-адреси до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2 ) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;


{ перетворення  IP-адреси  в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;


{ перетворення  крапкову десяткову IP-адресу  в мережний байт типу  DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;


{ перетворення  номер порту в мережний байт типу  DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;
```

```
{ перетворення  номера порту в мережний байт типу  string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;


{ перетворення  відомого номера порту у сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;


{ фіксуємо параметри досліджуваної мережі з неоднорідним трафіком }

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' HOSTNAME          : '  + HostName );
        List.Add( ' DOMAIN            : '  + DomainName );
        List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP SCOPE        : '  + ScopeID );
        List.Add( ' ROUTING ENABLED   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   ENABLED   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     ENABLED   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
            List.Add( ' DNS SERVER ADDR   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         //
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   //
begin
    InfoSize := 0 ;   //
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  //
    if result <> ERROR_BUFFER_OVERFLOW then exit ; //
    GetMem (FixedInfo, InfoSize) ;                    //
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   //
```

```
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  //
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                        NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                        PDnsServer^.IPAddress ;  //
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                        Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    //
    end ;
end;

//---------------------------------------------------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : ' + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//---------------------------------------------------------------------------

// включаємо байти вводу/виводу для кожного адаптеру

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize  : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; //
  TableSize := 0;
   //  перший виклик: беремо необхідний розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  //
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищуємо буфер, з W98 не потрібно

   // беремо показчик на таблицю
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
```

```
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // перетворюємо
Unicode у string
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
               ' %0.8x - %3d - %16s - %8d - %12d - %2d - %2d - %10d - %10d  -
%-s - %-s' ,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
counters are 32-bit
               sIfName, sDescr] )  //  , додаємо введення/вивід
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // звільняємо пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищуємо буфер, з W98 не
потрібно
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//---------------------------------------------------------------------------
{ інформація про мережні адаптери }
```

```
function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                                  AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та масок для IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].IPAddressList) <= I then
                begin
                     SetLength (AdpRows [AdpTot].IPAddressList, I * 2) ;
                     SetLength (AdpRows [AdpTot].IPMaskList, I * 2) ;
                end ;
            end ;
```

```
            AdpRows [AdpTot].IPAddressTot := I ;

        // беремо список IP адрес для GatewayList
            I := 0 ;
            PIpAddr := @AdapterInfo^.GatewayList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].GatewayList) <= I then
                            SetLength (AdpRows [AdpTot].GatewayList, I * 2) ;
            end ;
            AdpRows [AdpTot].GatewayTot := I ;

        // беремо список IP адрес для GatewayList
            I := 0 ;
            PIpAddr := @AdapterInfo^.DHCPServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].DHCPServer) <= I then
                            SetLength (AdpRows [AdpTot].DHCPServer, I * 2) ;
            end ;
            AdpRows [AdpTot].DHCPTot := I ;

        // беремо список IP адрес для PrimaryWINSServer
            I := 0 ;
            PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                            SetLength (AdpRows [AdpTot].PrimWINSServer, I * 2)
;
            end ;
            AdpRows [AdpTot].PrimWINSTot := I ;

       // беремо список IP адрес для SecondaryWINSServer
            I := 0 ;
            PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                            SetLength (AdpRows [AdpTot].SecWINSServer, I * 2) ;
            end ;
            AdpRows [AdpTot].SecWINSTot := I ;

            AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
            AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

            inc (AdpTot) ;
            if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot * 2) ;  // більше пам'яті
            AdapterInfo := AdapterInfo^.Next;
         end ;
         SetLength (AdpRows, AdpTot) ;
      end ;
  finally
      FreeMem( pBuf );
  end ;
end ;
```

```
procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;  jpt
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' -'  + Description ); // jpt : не корисне
            List.Add( Format(' %8.8x - %6s - %16s - %2d - %16s - %16s - %16s' ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : not useful
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /'  + IPMaskList [J] + '
- '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   '  ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//---------------------------------------------------------------------------
{ зчитуємо кількість раундів, та час звертання до IP }
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT := -1; //
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;

//---------------------------------------------------------------------------
{ ARP-таблиця – список відношень між віддаленими IP та віддаленими MAC-адресами.

}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
```

```
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  //  перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   //
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then //.
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x - %12s - %16s - %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we _must_ restore pointer!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//------------------------------------------------------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  //  перший виклик : беремо розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  //
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
```

```
    EXIT;

  // беремо розмір пам'яті, який потрібно та здійснюємо виклик знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s -> %15s : %-7s -> %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) = -1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------------------------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats     : TMibTCPStats;
  ErrorCode    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі: '  + TCPToAlgo[dwRTOAlgorithm] );
      List.Add( ' Мінімальний час виведення         : '  + IntToStr( dwRTOMin )
+ '  ms'  );
      List.Add( ' Максимальний час виведення         : '  + IntToStr( dwRTOMax )
+ '  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
```

```
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Відновлений зв'язок   : '  + IntToStr( dwEstabResets ) );
      List.Add( ' Поточний зв'язок .: '  + IntToStr( dwCurrEstab ) );
      List.Add( ' Отримано сегментів        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлено сегментів            : '  + IntToStr( dwOutSegs )
);
      List.Add( ' Ретрансльовано сегментів   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилки входження          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виведення          : '  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки   : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-------------------------------------------------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  //  перший виклик : беремо розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо потрібний розмір пам'яті, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо наступний запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
```

```
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  //  перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  //
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we must restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-------------------------------------------------------------------------
{ беремо дані з таблиці маршрутизатора Cisco}
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
```

```
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  //  перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                  dwForwardType := 1 ;   //  , враховуємо погані значення
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблоковане пересилання      : '  + ' Так'  )
```

```
      else
        List.add( ' Розблоковане пересилання      : '  + ' Hi'  );
      List.add( ' Вбудований TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримані датаграми       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку     (в)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Адреса помилкова    (в)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
//
      List.add( ' Невідомий протокол (в)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Датаграма відвергнута     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Запит виведення           : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрутизація в маршрутизаторі Cisco відвергнута     : ' +
inttostr( dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (з)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Час виведення перебору      : '  + inttostr( dwReasmTimeOut )
);
      List.add( ' Запити перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Перебори здійснено : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Фрагментація мережі з неоднорідним трафіком успішна: '  +
inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграми фрагментовано    : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' маршрут в таблиці маршрутизації Cisco : '  + inttostr(
dwNumRoutes ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      //
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//------------------------------------------------------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats     : TMibUDPStats;
  ErrorCode    : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграмми (в)     : '  + inttostr( dwInDatagrams ) );
      List.add( ' Датаграмми (з)    : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів           : '  + inttostr( dwNoPorts ) );
      List.add( ' Промилка    (в)     : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     //
```

```
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------------------------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Отримано повідослень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилки ICMP              : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування    : '  + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищено        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми параметрів      : '  + IntToStr( dwParmProbs ) );
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перенаправлення           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу  : '  + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адреси : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор маски адреси  : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення відправлено        : '  + IntToStr( dwMsgs )
);
      ICMPOut.Add( 'Помилки ICMP            : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування    : '  + IntToStr( dwDestUnreachs ) );
      ICMPOut.Add( 'Час перевищено        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми параметрів      : '  + IntToStr( dwParmProbs ) );
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перенаправлення           : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит маски адреси: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор маски адреси   : '  + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//-----------------------------------------------------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;
```

```
finalization
  RecentIPs.Free;
end.
```

Файл About.pas - довідка

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```