Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення для моніторингу мережевої активності клієнтів»

Студента 4 курсу, 13 групи,
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Шульга Євгеній Григорович

Науковий керівник
кандидат фізико-математичних наук _____ *підпис керівника* Філімонова Тетяна Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«20» грудня 2020р.

Завдання
на випускну кваліфікаційну роботу (проект) студенту

Шульга Євгеній Григорович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)

«Розробка програмного забезпечення для моніторингу мережевої активності клієнтів»
Затверджена наказом ректора від *«04» грудня 2019 р. № 4111*
2. Строк здачі студентом закінченої роботи *29 травня2020 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: обґрунтування та розробка програмного забезпечення для моніторингу мережевої активності клієнтів
Об'єкт дослідження: процес розробки програмного забезпечення для моніторингу активності мережі
Предмет дослідження: інформаційні системи і технології в системі управління мережами
4. Перелік графічного матеріалу_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пурський О.І.	12.12.2020 р.	12.12.2020 р.
2	Пурський О.І..	12.12.2020 р.	12.12.2020 р.
3	Пурський О.І.	12.12.2020 р.	12.12.2020 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Огляд існуючих систем моніторингу та аналізу мережевого трафіку

1.1. Обґрунтування необхідності моніторингу та контролю мережевої активності

1.2. Аналіз сучасних систем моніторингу мережевої активності.

1.3. Класифікація засобів моніторингу

РОЗДІЛ 2. Організація розробки програмного забезпечення для моніторингу мережевої активності

2.1. Функціональні вимоги до програмного забезпечення

2.2. Модель аналізу мережевої активності на основі класифікації мережевого трафіку

2.3. Алгоритм аналізу мережевого трафіку

РОЗДІЛ 3. Розробка програмного забезпечення для аналізу мережевої активності

3.1. Підбір параметрів та програмна реалізація проекту

3.2. Розробка графічного інтерфейсу

3.3. Особливості роботи та можливості масштабування програмного забезпечення

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*05.10.2020*	*05.10.2020*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*18.12.2020*	*18.12.2020*
3	*Вступ*	*03.02.2021*	*03.02.2021*
4	*РОЗДІЛ 1. Огляд існуючих систем моніторингу та аналізу мережевого трафіку*	*26.02.2021*	*26.02.2021*
5	*РОЗДІЛ 2. Організація розробки програмного забезпечення для моніторингу мережевої активності*	*06.04.2021*	*06.04.2021*
6	*РОЗДІЛ 3. Розробка програмного забезпечення для аналізу мережевої активності*	*12.05.2021*	*12.05.2021*
7	*Висновки*	*15.05.2021*	*15.05.2021*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*20.05.2021*	*20.05.2021*
9	*Попередній захист випускної кваліфікаційної роботи*	*26.05.2021*	
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*27.05.2021*	
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*31.05.2021*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «22» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи (проекту)
Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Шульга Є.Г
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)
_____ 31.05.2021 р.
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми _____ Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.
(підпис, прізвище, ініціали)
«_____»_____2021 р.

АНОТАЦІЯ

бакалаврської дипломної роботи Шульга Євгеній Григорович
на тему «Розробка програмного забезпечення для моніторингу мережевої активності».

Метою роботи є розробка програмного забезпечення для моніторингу мережевої активності клієнтів.

Дослідження особливості розробки та функціонування програмних забезпечень подібного функціоналу. Проведено порівняння та обрано мову програмування Delphi.

У ході виконання даної дослідницької роботи було розроблено програмний код та графічний інтерфейс програмного забезпечення для моніторингу мережевої активності клієнтів.

Загальний обсяг роботи: 50 сторінок, 7 рисунків, 12 таблиця, 25 посилання.

Ключові слова: моніторинг мережевої активності, програмне забезпечення, Delphi.

ANNOTATON

Bachelor`s degree of Shulha Yevhenii
entitled "Development of software for monitoring network activity"

The aim of the work is to develop software for monitoring network activity.

Research of features of development and functioning of software of similar functionality. A comparison was made and the Delphi programming language was selected.

In the course of this research work, software code and graphical interface of software for monitoring network activity were developed.

Total volume of work: 50 pages, 7 figures, 12 tables, 25 links.

Keywords: network activity monitoring, software, Delphi.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

IEEE — Institute of Electrical and Electronic Engineers (Інститут інженерів з електротехніки та електроніки) - організація, яка розробляє і публікує стандарти.

LAN — Local Area Network - локальна мережа, ЛВС. адреса - Media Access Control - ідентифікаційний номер мережевого пристрою, який визначається, як правило, виробником.

TCP / IP — Transmission Control Protocol / Internet Protocol - протокол управління передачею / протокол Internet.

ВСТУП

Моніторинг мережевої активності – це складне завдання, яке потребує великої кількості ресурсів серверу. Моніторинг та аналіз трафіку необхідні для того, щоб ефективніше діагностувати та вирішувати проблеми, не доводячи мережеві сервіси до простою протягом тривалого часу.

З використанням, орієнтованих та не орієнтованих на маршрутизатори відомих методів моніторингу, з'являються потреби в оновленні програмного забезпечення, яке змогло збільшити ефективність моніторингу при використанні не найсучаснішого обладнання.

На противагу їм є методи, що не ґрунтуються на маршрутизаторах, але вимагають встановлення спеціального апаратного та програмного забезпечення. Методи моніторингу тісно пов'язані зі завданнями забезпечення ефективного завантаження обладнання, безперебійної роботи мережі та запобігання несанкціонованим атакам на мережу.

Актуальність дослідження полягає в тому, що створення власного програмного продукут моніторингу активності системи, допоможе отримати максимальній результат від аналітики та зменшити енерговитирати комп'ютерів.

Під *об'єктом* дослідження розуміємо процес проектування та розробки програмного забезпечення для моніторингу мережевої активності.

Предметом дослідження постають технології створення системи моніторингу мережевої активності клієнтів.

Мета дослідження – полягає в створенні програмного забезпечення для моніторингу мережевої активності клієнтів.

Для досягнення поставленої мети дослідження, головними завданнями є:

- аналіз вже існуючих програмих забезпечень монтіторингу мереж;
- обрання оптимального методу створення програмного забезпечення моніторингу мережевої активності;
- дослідження сучасних методів розробки програмних забезпечень;
- проектування та розробка програмного забезпечення.

Методи дослідження: При вирішенні поставленого завдання використовувалися наукові досягнення в областях розробки інформаційних систем і програмного забезпечення

- Для практичного вирішення поставлених задач використовувалися такі методи:
- статистичні методи обробки;
- метод модульного проектування;
- методи класифікації даних;
- методи кластеризації даних.

РОЗДІЛ 1 Огляд існуючих систем моніторингу та аналізу мережевого трафіку

1.1 Обґрунтування необхідності моніторингу та контролю мережевої активності

Постійний контроль за роботою локальної мережі становить основу будь-якої корпоративної мережі, що необхідний для підтримки її в працездатному стані. Контроль - це важливий перший етап, який повинен виконуватися при управлінні мережею. Зважаючи на важливість цієї функції її часто відокремлюють від інших функцій систем управління і реалізують спеціальними засобами. Такий поділ функцій контролю і власне управління, корисно для невеликих і середніх мереж, для яких установка інтегрованої системи управління, економічно недоцільна. Використання автономних засобів контролю допомагає адміністратору мережі виявити проблемні ділянки мережі, а їх відключення або реконфігурацію він може виконувати в цьому випадку вручну [1]. Процес контролю роботи мережі зазвичай ділять на два етапи - моніторинг та аналіз.

На етапі моніторингу виконується процедура збору первинних даних про роботу мережі: статистика про кількість циркулюючих в мережі кадрів і пакетів різних протоколів, стан портів концентраторів, комутаторів і маршрутизаторів.

Далі виконується етап аналізу, під яким розуміється складніший і інтелектуальний процес осмислення зібраної на етапі моніторингу інформації, зіставлення її з даними, отриманими раніше, і вироблення припущень про можливі причини сповільненої або ненадійної роботи мережі.

Завдання моніторингу вирішуються програмними і апаратними вимірниками, тестерами, мережевими аналізаторами, вбудованими засобами моніторингу комунікаційних пристроїв, а також агентами систем управління. Завдання аналізу вимагає більш активної участі людини і використання таких складних засобів, як експертні системи, що акумулюють практичний досвід багатьох мережевих фахівців [2].

1.2 Аналіз сучасних систем моніторингу мережевої активності

Платформа для системного моніторингу мереж повинна мати наступні властивості: маштабованість, розподіленості відповідно до концепції клієнта / сервера та відкритість.

Перші дві властивості тісно пов’язані. Хороша масштабність приймається за рахунок розподілу систем управління. Поширеність тут означає, що система може включати кілька серверів та клієнтів. Сервери збирають дані про поточний стан мережі від агентів (SNMP, CMIP або RMON), вбудованих в обладнання мереж, і накопичують їх у своїй базі даних. Клієнти являють собою графічні консолі, для яких працюють адміністратори мереж. Програмне забезпечення клієнтської системи управління, приймає запрошення на виконання яких-небудь дій від адміністратора (наприклад, побудова детальної картки частин мережі) та обробляється за необхідною інформацією для серверів [3]. Сервер володіє потрібною інформацією та передає її клієнту, якщо ні - він намагається зібрати її від агентів.

Ранні версії програмного забепечення системного управління відповідали всім функціям в одному комп'ютері, за яким працював адміністратор. Для невеликих мереж або мереж з невеликим обсягом керованого обладнання, структура моніториться в повному обсязі, але при великому обсязі керованого обладнання, єдиним комп'ютером до якого надходить інформація від усіх пристроїв мереж, вважається слабким місцем [4]. Але це тільки у випадку, якщо програмне забезпечення справляється з більшим потоком даних, а сам комп'ютер не здатний їх обробити.

До найпопулярніших продуктів мережевого управління відносяться чотири системи: Spectrum компанії CabletronSystems, OpenView фірми Hewlett-Packard, NetView корпорації IBM та Solstice виробництва SunSoft - лпідрозділ SunMicrosystems. Три компанії з чотирьох виробляють комунікаційне обладнання. Система Spectrum найкраще керує обладнанням компанії Cabletron, OpenView - обладнанням компанії Hewlett-Packard та NetView - обладнанням компанії IBM.

При побудові мережевих карт, що знаходяться на устаткуванні інших виробників, ця система починає помилятися і приймає одне обладнання за інше. Для виправлення цього недоліку розробники системи управління включають підтримку не тільки стандартних баз MIBI, MIBII та RMONMIB, але і багаторазових приватних фірм-виробників MIB. Лідер в цій області - система Spectrum, що підтримує близько 1000 баз MIB різних виробників [5].

До найбільш відомих систем відносяться компанії Optivity BayNetworks, CiscoWorks компанії CiscoSystems, Transcend компанії 3Com. Система Optivity, дозволяє моніторити систему та керувати мережами, комутаторами та концентраторами компаній BayNetwork, повністю використовуючи всі їх можливості та властивості. Оснащення інших виробників підтримується на рівні базових функцій управління. Система Optivity працює на платформі OpenView компанії Hewlett-Packard і SunNetManager компанії SunSoft. Однак, робота на основі будь-якої платформи управління з декількома системами, такими як Optivity, занадто складна і потрібна, для комп'ютерів, на яких все це буде працювати, має дуже потужні процесори та велику оперативну пам'ять.

1.3 Класифікація засобів моніторингу

Все різноманіття засобів, що застосовуються для аналізу і діагностики мереж, можна розділити на кілька великих класів:

- Агенти систем управління, що підтримують функції однієї зі стандартних MIB (Management Information Base) - база даних інформацією управління, яка використовується в процесі управління мережею в якості моделі керованого об'єкта в архітектурі агент-менедже і постачають інформацію по протоколу SNMP або CMIP. Для отримання даних від агентів зазвичай потрібна наявність системи управління, яка щороку збирає дані від агентів в автоматичному режимі [6].
- Вбудовані системи діагностики і управління (Embedded systems). Ці системи виконуються у вигляді програмно-апаратних модулів, що встановлюються в

комунікаційне обладнання, а також у вигляді програмних модулів, вбудованих в операційні системи. Вони виконують функції діагностики і управління тільки одним пристроєм, і в цьому їх основна відмінність від централізованих систем управління. Прикладом засобів цього класу може служити модуль управління Багатосегментний повторителем Ethernet, який реалізує функції автосегментацією портів при виявленні несправностей, приписування портів внутрішнім сегментам повторювача і деякі інші [7]. Як правило, вбудовані модулі управління «за сумісництвом» виконують роль SNMP-агентів, що поставляють дані про стан пристрою для систем управління.

- Аналізатори протоколів (Protocol analyzers). Являють собою програмні або апаратно-програмні системи, які обмежуються на відміну від систем управління лише функціями моніторингу і аналізу трафіку в мережах. Хороший аналізатор протоколів може захоплювати і декодувати пакети великої кількості протоколів, що застосовуються в мережах. Зазвичай кілька десятків. Аналізатори протоколів дозволяють встановити деякі логічні умови для захоплення окремих пакетів і виконують повне декодування захоплених пакетів, тобто показують в зручній для фахівця формі вкладеність пакетів протоколів різних рівнів один в одного з розшифровкою змісту окремих полів кожного пакета.
- Експертні системи. Цей вид систем акумулює знання технічних фахівців про виявлення причин аномальної роботи мереж і можливі способи приведення мережі в працездатний стан. Експертні системи часто реалізуються у вигляді окремих підсистем різних засобів моніторингу та аналізу мереж: систем управління мережами, аналізаторів протоколів, мережевих аналізаторів. Найпростішим варіантом експертної системи є контекстно-залежна система допомоги. Більш складні експертні системи являють собою, так звані бази знань, що володіють елементами штучного інтелекту. Прикладами таких систем є експертні системи, вбудовані в систему управління Spectrum компанії Cabletron і аналізатора протоколів Sniffer компанії Network General.

Робота експертних систем полягає в аналізі великого числа подій для видачі користувачеві короткого діагнозу про причини несправності мережі [8].

Устаткування для діагностики і сертифікації кабельних систем. Умовно це устаткування можна поділити на чотири основні групи: мережеві монітори, прилади для сертифікації кабельних систем, кабельні сканери і тестери.

- Мережеві монітори (звані також мережевими аналізаторами) призначені для тестування кабелів різних категорій. Мережеві монітори збирають також дані про статистичні показники трафіку - середньої інтенсивності загального трафіку мережі, середньої інтенсивності потоку пакетів з певним типом помилки і т. П. Ці пристрої є найбільш інтелектуальними пристроями з усіх чотирьох груп пристроїв даного класу, так як працюють не тільки на фізичному , але і на канальному, а іноді і на мережевому рівнях.
- Пристрої для сертифікації кабельних систем виконують сертифікацію відповідно до вимог одного з міжнародних стандартів на кабельні системи.
- Кабельні сканери використовуються для діагностики мідних кабельних систем.
- Тестери призначені для перевірки кабелів на відсутність фізичного розриву. Багатофункціональні портативні пристрої аналізу і діагностики. У зв'язку з розвитком технології великих інтегральних схем з'явилася можливість виробництва портативних приладів, які поєднували б функції декількох пристроїв: кабельних сканерів, мережних моніторів і аналізаторів протоколів [9].

Висновки розділу

Проведено аналіз та класифікацію систем моніторингу, що дало змогу обрати найефективніші параметри для майбутнього програмного забезпечення.

РОЗДІЛ 2 Організація розробки програмного забезпечення для моніторингу мережевої ативності

2.1 Функціональні вимоги до програмного забезпечення

Існує велика кількість додатків адміністрування, які можна використовувати для моніторингу мережі. Всі вони розрізняються по інтерфейсу та протоколами що використовуються.

Адміністрування надає повний контроль над комп’ютером в мережі: воно дає можливість дистанційно керувати робочим столом комп'ютера, можливість копіювання або видаляти файлів, запускати додатки і т. д.

Даний програмний продукт повинен підходити під будь-які системні характеристики, а саме працювати в максимальній продуктивності на процесорах Intel Core 2 Duo з тактовою частотою 2.2 Ггц або аналогічних, при обмеженні в 512 МБ оперативної пам'яті і займати на жорсткому диску простір, що не заважає роботі операційної системи.

Програма має працювати в багатопотоковому режимі для його максимальної продуктивності при великій кількості підключених комп’ютерів.

Клієнтська частина повинна працювати у фоновому режимі і бути захищена від несанкціонованого зміни налаштувань програми.

Потрібно врахувати, що додаток буде використовуватися користувачами з різним рівнем знань в сфері інформаційних технологій. Тому програмний продукт повинен володіти зручним призначеним для користувача інтерфейсом.

Перед тим як вибрати засоби розробки необхідно провести їх порівняльний аналіз. Для порівняння були взяті такі мови програмування: Delphi, C ++, C #, Java.

Delphi - імперативний, структурований, об'єктно-орієнтована мова програмування, діалект Object Pascal. Починаючи з середи розробки Delphi 7.0, в офіційних документах Borland стала використовувати назву Delphi для позначення мови Object Pascal. Починаючи з 2007 року вже мова Delphi (похідний від Object Pascal) почав жити своїм самостійним життям і зазнавав різні зміни, пов'язані з

сучасними тенденціями (наприклад, з розвитком платформи .NET) розвитку мов програмування: з'явилися class helpers, перевантаження операторів.

Спочатку середовище розробки Delphi була призначена виключно для розробки додатків Microsoft Windows, потім був реалізований варіант для платформ Linux (як Kylix), однак після випуску в 2002 році Kylix 3 його розробка була припинена, і незабаром було оголошено про підтримку Microsoft .NET [10].

Реалізація середовища розробки проектом Lazarus (Free Pascal, компіляція в режимі сумісності з Delphi) дозволяє використовувати його для створення додатків на Delphi для таких платформ, як Linux, Mac OS X і Windows CE.

Також робилися спроби використання мови в проектах GNU (наприклад, Notepad GNU) і написання компілятора для GCC.

C++ - компільований статично типізований мова програмування загального призначення.

Підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції. Стандартна бібліотека включає, в тому числі, загальновживані контейнери і алгоритми. C++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником - мовою C, - найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування [11].

Будучи одним з найпопулярніших мов програмування C++ широко використовується для розробки програмного забезпечення. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також розважальних програм. Існує безліч реалізацій мови C++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C ++, Intel C ++ Compiler, Embarcadero (Borland) C++ Builder і інші. C++ зробив величезний вплив на інші мови програмування, в першу чергу на Java і C #.

Java - об'єктно-орієнтована мова програмування, розроблений компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Програми Java зазвичай компілюються в спеціальний байт-код, тому вони можуть працювати на будь-який віртуальної Java-машині (JVM) незалежно від комп'ютерної архітектури [12].

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером) викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять те, що виконання байт-коду віртуальною машиною може знижувати продуктивність програм і алгоритмів, реалізованих на мові Java.

C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML [12].

Перейнявши багато від своїх попередників - мов C++, Delphi, Модула, Smalltalk і особливо Java - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне спадкування класів. Виходячи з вищесказаного мова Delphi є найоптимальнішою для створення даного програмного забезпечення.

2.2 Модель аналізу мережевої активності на основі класифікації мережевого трафіку

Виділяють три основні варіанти класифікації. Далі вони перераховані в порядку збільшення «Точності» класифікації. Тип трафіку не є достатньо змістовним способом класифікації і, як правило, або не береться подальшого аналізу, або піддається досить простий додаткової уточнюючої класифікації. Залежно від сфери застосування, типи можуть бути різними.

Серед прикладів, можна вказати: P2P, відео-стрімінг, веб-трафік - в разі систем збору статистики і моніторингу, трафік мережевої атаки / нормальний трафік - в разі систем захисту від мережевих атак, трафік, що містить / який не містить об'єкти копірайту, в випадку систем контролю копірайту. Використовує протокол прикладного рівня (protocol identification) є досить змістовним і може, як використовуватися безпосередньо - наприклад, в системах збору статистики і моніторингу для підвищення рівня точності [13].

Основним способом подальшої обробки є розбір протоколу, що включає дві основних функції - складання сесії прикладного рівня, в разі необхідності вилучення даних протоколу з окремих його полів (метаінформація рівня протоколу).

Додаток, передає дані (application identification), дає максимально деталізований рівень класифікації. На цьому рівні можуть здійснюватися ті ж види обробки, що і на рівні протоколу прикладного рівня, а також вилучатись і інтерпретуватися дані (метаінформація) конкретного додатки, що відповідає більш високому рівню їх уявлення.

У різних прикладних задачах результати ідентифікації протоколів і додатків можуть інтерпретуватися і, відповідно піддаватися різної подальшій обробці (як і в разі ідентифікації типу трафіку).

Наприклад, в разі системи захисту від шкідливого коду, під протоколом може розумітися командний (command-and-control, C & C) протокол ботнету, а під додатком - конкретний вірус. Відповідно, яку видобувають метаінформація -

команди ботнету, передані їм дані, а мета аналізу - з'ясування його функціоналу, оцінка поширеності і дослідження можливостей його деактивації.

Вибір конкретної прикладної задачі може значно впливати як на вибір алгоритму класифікації, так і на його параметри і продуктивність.

У разі систем фільтрації за ключовими словами такий метод не підходить, так як в одному і тому ж мережевому потоці, в різних пакетах можуть зустрітися різні слова і, з точки зору системи класифікації, в цьому випадку даний потік потрапить відразу в кілька класів.

У загальному випадку, очевидно, що перший підхід набагато продуктивніше, так як доводиться аналізувати значно менші обсяги даних. Крім того, в ряді підходів, для додаткового прискорення, аналізують не все вміст пакета, а тільки деякий його префікс (за аналогією зі слайсинга). Наприклад, в роботі [14], для ідентифікації потоків, містять шифровані і стислі дані, використовуються тільки перші 16 байт пакетів.

В роботі [15] проведено оцінку впливу розміру аналізованого префікса пакету на точність класифікації по протоколам і швидкість роботи класифікатора на трьох знятих мережевих трасах Unibs-GT, Polito, Polito-GT.

Результати наведені на рис. 2.1, де на лівому графіку помилки класифікації позначені як misclassified, а трафік, який не вдалося класифікувати, як unknown.

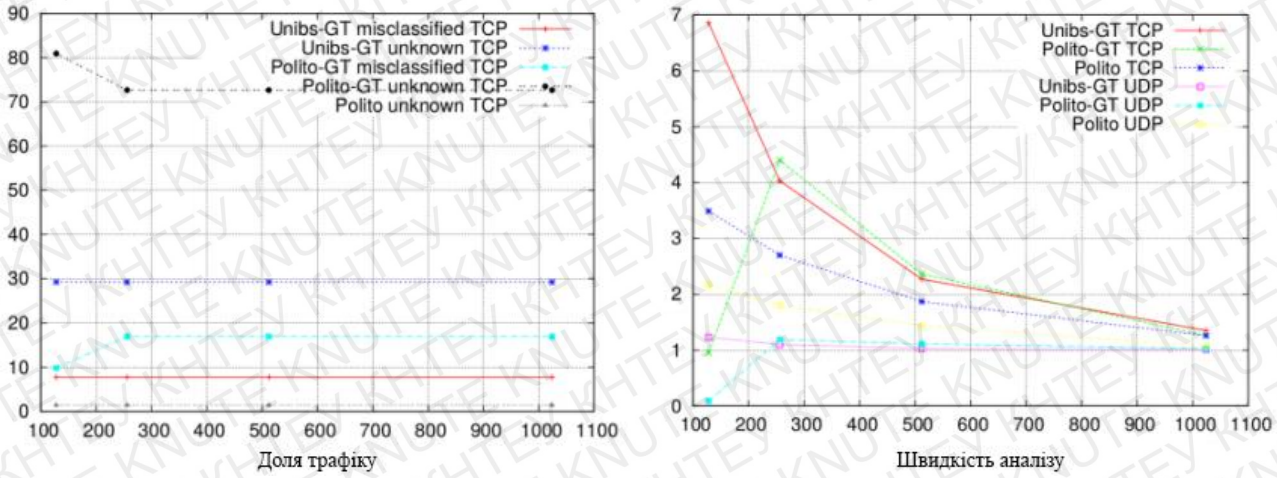


Рис. 2.1. Оцінка впливу довжини префікса на точність класифікації (зліва) і швидкість (праворуч).

На основі цих досліджень, зокрема робиться висновок про надмірності проведення IP-дефрагментації і TCP-нормалізації при вирішенні даної завдання, так як дані алгоритми досить ресурсомісткі і практично не впливають на точність. Це відбувається через те, що для класифікації, як правило, використовується не більше 256 байт пакетів, а мінімальний розмір фрагмента зазвичай не менше 576 байт. Тобто, для даного завдання PBFS підхід більш доцільний, ніж підхід MBFS.

Класичним підходом до класифікації є аналіз вмісту пакетів (payload-based). При цьому, як правило, виконується пошук «Сигнатур» - характерних ознак, які заздалегідь створюються для кожної програми або їх груп. Класифікація може виконуватися як на рівні окремих пакетів (stateless аналіз), або може враховуватися стан потоку (statefull аналіз). Для підвищення точності розпізнавання частина підходів використовує уточнені «сигнатури» на основі автоматів станів протоколів. При такому підході, одержувані повідомлення, після їх класифікації, зіставляються з переходами в різних автоматах протоколів, і оцінюється коректність послідовностей таких переходів. Ця група підходів називається Stateful Protocol Analysis Detection [16].

Історично, через брак обчислювальних потужностей, робилися спроби досягнення збільшення продуктивності алгоритму за рахунок вибору джерела даних, використовуваних алгоритмом в процесі класифікації, таким чином, щоб оброблювані дані, будучи не менш інформативними, ніж вміст пакетів, були б більш компактні. Ця група підходів (на відміну від «Сигнатурного») відноситься до класу «заснованих на виведення» (inference-based).

Одним з важливих переваг inference-based підходів є те, що якість аналізу не залежить від представлення даних в мережевих пакетах, в Зокрема, відсутні обмеження при аналізі стисненого / шифрованого трафіку. Далі будуть розглянуті основні підходи до вирішення завдання класифікації, їх особливості і обмеження застосовності.

Всі підходи на основі висновку можна розділити на групи за двома основними параметрам:

- що використовуються для виведення дані;

- використовуваний для їх аналізу алгоритм.

Всі види даних, в свою чергу, можна розділити на:

- характеристики окремих пакетів в рамках окремого потоку (Packet based);
- характеристики потоків в цілому (flow based).

До першої групи належать підходи, які використовують такі характеристики як: тимчасові проміжки між пакетами, послідовності розмірів пакетів [17], та ін.

До другої групи належать два основні підходи. Підхід на основі аналізу портів (port-based) при якому ідентифікація відбувається по одному з номерів портів потоку, на основі бази даних про характерні статичних портах, які використовують зареєстровані в IANA протоколи. Цей метод вважається малоефективним, тому що на даний момент існує велика кількість протоколів з динамічними номерами портів. Зокрема, до таких протоколів відносяться практично всі реалізації P2P. Крім того, часто використовуються схеми, при яких трафік деякого протоколу (наприклад, HTTP) передається по невластивому для нього номером порту (не 80 в разі HTTP).

Підходи на основі статистичної інформація про активність окремих хостів в мережі: в скількох і яких саме обмінах даними (потоках) брав участь даний хост, скільки даних, і в який бік передавалося і т.д. Ці дані зіставлялися з набором заздалегідь створених шаблонів різних видів серверів. Один з таких підходів описаний в роботі [18].

Алгоритми аналізу даних діляться на два основних напрямки:

- порівняння з тим чи іншим видом заздалегідь створеного шаблону;
- підхід на основі машинного навчання і подальшого розпізнавання.

2.3 Алгоритм аналізу мережевого трафіку

Загальна схема аналізу мережевого трафіку складається з наступної послідовності кроків, кожен з яких призводить до підвищення рівня уявлення об'єкта аналізу.

Захоплення пакетів, що проходять через контрольоване мережеве з'єднання. Результатом даного кроку є отримання об'єкта аналізу у вигляді мережевих пакетів.

Залежно від необхідної точності і швидкості подальшого аналізу, а також доступних обчислювальних потужностей можуть використовуватися різні підходи.

Слайсинга (slicing), при якому аналізу піддаються не весь вміст пакетів, а тільки деякий префікс (n перших байт). У ряді досліджень [19] показано, що цей підхід добре працює для подальшої класифікації трафіку по протоколам. В окремому випадку, якщо перехоплювати розмір дорівнює сумарному обсягу мережевих заголовків (L1-L3) є реалізацією технології SPI.

Самплинг (sampling), при якому перехоплюються не всі пакети, а тільки їх частина, яка може вибиратися за різними умовами, в залежності від потреб. В процесі розвитку технології було запропоновано велику кількість стратегій відбору [20].

Наприклад, для завдань моніторингу типів трафіку підходить варіант з вибором кожного n-го пакета (uniform sampling), де n може вибиратися в залежності від співвідношення ширини каналу і пропускної спроможності системи аналізу. завдання отримання інформації про повне стані мережі за результатами самплинга відома як inversion problem, зокрема, при застосуванні uniform sampling відбувається недооцінка середнього розміру пакетів, так як частіше будуть відбиратися пакети меншого розміру. Для передачі перехоплених даних використовується протокол PSAMP [21].

Для завдань, в яких потрібно максимально точний аналіз трафіку, наприклад для систем забезпечення мережевої безпеки, потрібно перехоплювати всі дані за все що надходить трафіку без втрат - для позначення цього підходу використовується термін lossless capture або deep packet capture (DPC).

Агрегирование пакетів в потоки за деякими адресним ознаками (Flow generaion [22]), отримання нового об'єкта для аналізу – мережевого потоку. Якщо при цьому дані пакетів надалі аналізі не враховуються, то такий вид аналізу називається «аналіз потоків» - flow based analysis (на відміну від packet-based аналізу, при якому аналізуються дані пакетів). На рис. 2.2 показані відмінності типових схем packet і flow-based аналізу. Flow-based аналіз широко використовується в силу значно менших вимог до потужності обчислювача і

пропускної здатності, за рахунок значного зниження обсягу даних для обробки. Такий вид аналізу може виконуватися як локально, так і віддалено від точки збору даних. Для передачі зібраних даних від точки збору до точки аналізу використовується велика кількість протоколів, частина з яких стандартизована у вигляді IPFIX [23], а частина розроблена окремими виробниками - Cisco NetFlow, Juniper Jflow.

В рамках підходу записи, що описують потік можуть містити різний набір даних.

Найбільш загальним набором таких даних є наступний:

- IP адреси джерела і адресата;
- протокол транспортного рівня;
- в разі протоколів TCP / UDP - номери портів джерела / адресата;
- набір лічильників: кількість переданих пакетів і байт, час створення і завершення потоку.

Виконання класифікації по протоколу прикладного рівня або конкретного мережевого додатком. Результатом даної операції є отримання нового об'єкта для аналізу - мережевого потоку конкретного протоколу або додатка (в цьому випадку пов'язаних потоків може бути кілька, наприклад, в разі VoIP додатки це потік і SIP і RTP). Після виконання даної операції можлива наступна додаткова обробка отриманого об'єкта, конкретний вид якої залежить від розв'язуваної прикладної задачі:

- розбір полів протоколу (protocol parsing);
- складання сесії протоколу для протоколів із встановленням з'єднання;
- вилучення даних додатка (content extraction) – сторінок сайтів (HTML), файлів різних типів (виконувані, зображення, текстові документи, і т.д.), електронних листів, аудіо-відео потоків і т.д.;
- розбір даних програми (application content parsing).

Рис. 2.2. Відмінності типових схем packet (зліва) і flow-based (праворуч) аналізу.

Модулі для накопичення, зберігання і обміну даними в форматі MIB реалізовані в більшості пристроїв. Передача даних здійснюється по протоколу SNMP [24]. Дані отримані таким шляхом мають низький обсяг і неспецифічні для протоколів. Наприклад, в рамках даного підходу, можна отримати відомості про загальну кількість пакетів і байт пройшли через конкретний мережевий інтерфейс конкретного мережевого пристрою.

Однією з причин розвитку MIB і flow-based підходів, незважаючи на їх порівняно низьку точність, послужила досі йде глобальна дискусія про законність і допустимості глибокого аналізу трафіку з точки зору порушення безпеки, прав на приватне життя і т.д.

На даний момент одним з наслідків даної дискусії є, зокрема, то, що в наукових роботах, трафік, який піддається глибокого аналізу попередньо проходить процедуру «анонімізації» за допомогою спеціальних коштів [25].

Висновки розділу

Завдяки аналізу технологій та дослідженню мов програмування, вдалося визначити як найзручніше реалізувати програмне забезпечення і було вибрано середовище розробки Delphi 7.

РОЗДІЛ 3 Розробка програмного забезпечення для аналізу мережевої активності

3.1 Підбір параметрів та програмна реалізація проекту

Беручи до уваги наведений огляд основних алгоритмів і схем аналізу мережевого трафіку можна сформулювати ряд функціональних і нефункціональних вимог, що пред'являються до сучасних систем аналізу мережевого трафіку. Всі вимоги можна розділити по підсистемах, до яким вони застосовуються і окремо виділити ті, які відносяться до всієї системі в цілому:

1. Система в цілому.
   - Підтримка масштабування по пропускній здатності аналізованого каналу передачі даних.
   - Мінімізація числа перестановок пакетів в рамках окремих потоків.
   - Можливість вбудовування додаткових коштів предобработки мережевих пакетів перед їх передачею підсистемі класифікації.
2. Підсистема перехоплення даних.
   - Підтримка розбору всіх протоколів нижче мережевого рівня, зустрічаються в контрольованому каналі (MPLS, VLAN і т.д.). Це необхідно, для забезпечення потрапляння всіх пакетів одного потоку в одну чергу обробки при виконанні балансування навантаження (змішування має виконуватися на рівні IP-пакета).
   - Використання кільцевого буфера для зберігання оброблюваних пакетів і режиму zero-copy, при наявності підтримки з боку мережевої карти, або 1-copy, при відсутності такої підтримки для економії ресурсів центрального процесора.
   - Для ефективного використання ресурсів багатопроцесорних і багатоядерних машин потрібна підтримка того чи іншого виду RSS-технології.

   Підсистема агрегації пакетів в потоки має наступні компонент:.

- Підтримка можливості завдання типу ключової інформації, по якої визначається приналежність пакета до потоку, для забезпечення гнучкості при використанні підсистеми для вирішення різних прикладних задач.
- Максимізація кількості одночасно оброблюваних потоків і часу життя кожного окремого потоку в умовах обмежених ресурсів пам'яті.
- Для обробки стислих даних необхідна можливість одночасного відстеження потоку, який представлений як в стислому, так і в розціпленого вигляді.

Підсистема класифікації має наступні компонент:

- Складність алгоритму пошуку сигнатур повинна бути не гірше ніж лінійної за вхідними даними «в середньому», а бажано і «в гіршому »випадку для стійкості системи до цілеспрямованим атакам.
- Розширюваний набір «сигнатур» для підтримки нових протоколів, їх груп і мережевих додатків.
- Хороша масштабованість по пам'яті при зростанні кількості «Сигнатур».
- Можливість попереднього поділу трафіку на «Прозорий» і «непрозорий» з метою зниження навантаження на дану підсистему.
- Можливість аналізу даних, представлених в різних кодуваннях.

Зважаючи на не, що програма буде створена в Delphi 7 в програмі буде використано об'ємні коди, які будут знаходитись в додатках.

Підключення до системи буде створено за рахунок настопного коду:

Таблиця 3.1

Підключення до системи

```
var
   NetShareEnum :function (pszServer        : PChar;
                           sLevel           : Cardinal;
                           pbBuffer         : PChar;
                           cbBuffer         : Cardinal;
                           pcEntriesRead,
                           pcTotalAvail     : Pointer   ):DWORD; stdcall;
```

- ServerName - повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо виконуємо у себе то даному параметру можна привласнити NIL.
- Level - повинен містити ідентифікатор структури.
- Bufptr - повинен містити адресу покажчика на масив структур.
- Prefmaxlen - якщо не ставити обмеження то даному параметру потрібно присвоїти DWORD.
- EntriesRead - повинен містити покажчик на змінну в яку запишеться кількість загальних ресурсів доступних на даний момент.
- Resume_handle - не використовується, повинен бути NIL.

Результати виконання будуть збережені в масиві структур переданих функції при її виклику. Використовується 6 типів структур переданих функції NetShareEnum:

- SHARE_INFO_0 - тільки Windows NT.
- SHARE_INFO_1 - тільки Windows NT.
- share_info_1 - тільки Windows 9x-Me.
- SHARE_INFO_2 - тільки Windows NT.
- share_info_50 - тільки Windows 9x-Me.
- SHARE_INFO_502 - тільки Windows NT.

Таблиця 3.2

Структура share_info_50

```
type
 TShareInfo50 = packed record
  shi50_netname        : array [0..12] of Char;
  shi50_type           : Byte;
  shi50_flags          : Word;
  shi50_remark         : PChar;
  shi50_path           : PChar;
  shi50_rw_password    : array [0..8] of Char;
  shi50_ro_password    : array [0..8] of Char;
 end;
```

- shi50_netname - містить рядок містить мережеве ім'я ресурсу.

- shi50_type - визначає тип ресурсу (докладніше в MSDN).
- shi50_flags - містить інформацію про права доступу до ресурсу.
- shi50_remark - покажчик на рядок містить необов'язковий коментар до ресурсу.
- shi50_path - містить локальне розташування ресурсу.
- shi50_rw_password - містить пароль на запис – читання.
- shi50_ro_password - містить пароль на читання.

Реально отримати значення двох останніх полів можна тільки при отриманні інформації про своєму комп'ютері, в інших випадках вони залишаються порожніми.

Таблиця 3.3

Структура SHARE_INFO_2

```
type
 TShareInfo2 = packed record
  shi2_netname                : PWChar;
  shi2_type                   : DWORD;
  shi2_remark                 : PWChar;
  shi2_permissions            : DWORD;
  shi2_max_uses               : DWORD;
  shi2_current_uses           : DWORD;
  shi2_path                   : PWChar;
  shi2_passwd                 : PWChar;
 end;
 PShareInfo2 = ^TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^TShareInfo2Array;
```

Єдина відмінність функцію для NT перейменовано в NetShareEnumNT, для того щоб не було двох однакових функцій з різними параметрами.

Реалізується виклик функцій. Спочатку потрібно визначиться, під якою системою працює програма, щоб дізнатися яку частину коду (для NT чи ні) використовувати в даний момент. Для цього створюється невелику функція, яка і визначатиме тип системи.

Таблиця 3.4

Визначення типу системи

```
function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then
  begin
    Result := False;
    Exit;
  end else
    Result := True;
  case Ver.dwPlatformId of
    VER_PLATFORM_WIN32_NT      : Value := True;   //Windows NT
    VER_PLATFORM_WIN32_WINDOWS : Value := False;  //Windows 9x-Me
    VER_PLATFORM_WIN32s        : Result := False; //Windows 3.x
  end;
end;
```

Після визначення типу системи, а ця функція найголовнішою, так как програма визначає середовище роботи, потрібно перевірти звязки. Якби використовувались статичне зв'язування і заздалегідь визначили бібліотеки і імена бібліотечних функцій, гарантовано отримали б в процесі виконання помилку, так як бібліотека SVRAPI відсутня в Windows NT, а NETAPI32 відсутня в Windows 9x-Me. Для цього потрібно після визначення типу системи завантажити необхідну бібліотеку, і вже тільки після цього зв'язати функції з бібліотечними.

Таблиця 3.5

Отримуємо адреси функцій

```
procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;
```

Продовження табл. 3.5

```
entriesread,totalentries:DWORD;
  Share : array [0..512] of TShareInfo50;
  pcEntriesRead,pcTotalAvail:Word;
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close;
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareEnumNT := GetProcAddress(FLibHandle,'NetShareEnum');
    if not Assigned(NetShareEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil;
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
          @entriesread,@totalentries,nil) <> 0 then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then
    for i:= 0 to entriesread - 1 do
          lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareEnum := GetProcAddress(FLibHandle,'NetShareEnum');
    if not Assigned(NetShareEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
          @pcEntriesRead,@pcTotalAvail) <> 0 then
    begin
      FreeLibrary(FLibHandle);
      Exit;
```

Продовження табл. 3.5

```
end;
    if pcEntriesRead > 0 then
    for i:= 0 to pcEntriesRead - 1 do
            lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle);
end;
```

При виконанні цього коду ListBox заповниться назвами загальних ресурсів, які беруться з масиву структур переданих функції. Зазвичай в NT використовуються функції NetApiBufferAllocate і NetApiBufferFree - за допомогою них виділяється і звільняється пам'ять під масив структур.

Отримувати список загальних ресурсів ви навчилися. Тепер розглянемо функцію NetShareDel яка дозволить нам вибраним загальний ресурс.

Таблиця 3.6

Закриття локального ресурсу

```
var
              NetShareDel:function  (pszServer,
                                     pszNetName      :PChar;
                                     usReserved      :Word ): DWORD; stdcall;
```

Отже, додавши до програми ще одну кнопку і назвавши її btnCloseShares. оброблювач цієї кнопки буде містити код, який визначає поточний вибраний елемент в списку поточних загальних ресурсів, і буде передавати його ім'я (Strings [xx]) в якості другого параметра функції (дод А).

Таблиця 3.7

Відкриття локального ресурсу

```
var
              NetShareAdd: function (       pszServer       :Pchar;
                                            SLevel          :Cardinal;
                                            PbBuffer        :PChar;
                                            CbBuffer        :Word):DWORD;
stdcall;
```

- servername - повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо відкриваємо локальний ресурс то даному параметру потрібно присвоїти NIL.
- level - повинен містити ідентифікатор структури.
- buf - повинен містити покажчик на структуру.
- parm_err - містить покажчик члена структури викликає помилку

Функції для NT також використовується покажчик а не адресу покажчика. Обидві функції використовують структури описані вище. Для демонстрації їх використовується share_info_50 і SHARE_INFO_502 (дод Б). В обробник OnClick цієї кнопки поміщено код, що дозволяє вибрати папку, загальний доступ до якої ми хочемо відкрити.

Таблиця 3.8

Діалог вибору потоку сервера

```
function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := 'Specify a directory';
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := '';
  Result := String(TempPath);
end;
```

Для визначення адміністраторів, що працюють в мережі використовують функцію NetSessionEnum.

Таблиця 3.9

Визначення адміністраторів

```
var
              NetSessionEnum:function(      pszServer       : PChar;
                                            sLevel          : DWORD;
                                            pbBuffer        : Pointer;
                                            cbBuffer        : DWORD;
                                            pcEntriesRead,
                                            pcTotalAvial    : Pointer):integer;
stdcall;
```

- ServerName - повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція.
- UncClientName - містить покажчик на рядок містить ім'я сесії про яку ми хочемо отримати інформацію.
- UserName - містить покажчик на рядок містить ім'я користувача про який ми хочемо отримати інформацію, якщо потрібно переглянути все, параметру потрібно присвоїти NIL.
- Level - повинен містити ідентифікатор структури.
- Bufptr - повинен містити адресу покажчика на масив структур.
- Prefmaxlen - повинен містити максимальну довжину повернутих даних в байтах, якщо не ставити обмеження то даному параметру потрібно присвоїти DWORD.
- Entriesread - повинен містити покажчик на змінну в яку запишеться кількість загальних ресурсів доступних на даний момент.
- Totalentries - не використовується
- Resume_handle - не використовується, повинен бути NIL

Існує також 6 типів структур переданих функції NetSessionEnum.

- SESSION_INFO_0 - тільки Windows NT.
- SESSION_INFO_1 - тільки Windows NT
- SESSION_INFO_2 - тільки Windows NT.
- SESSION_INFO_10 - тільки Windows NT.

- SESSION_INFO_502 - тільки Windows NT.
- session_info_50 - тільки Windows 9x-Me.

Таблиця 3.10

Структура session_info_50

```
type
 TSessionInfo50 = packed record
   sesi50_cname       : PChar;
   sesi50_username    : PChar;
   sesi50_key         : Cardinal;
   sesi50_num_conns   : Word;
   sesi50_num_opens   : Word;
   sesi50_time        : Cardinal;
   sesi50_idle_time   : Cardinal;
   sesi50_protocol    : Byte;
   pad1               : Byte;
 end;
```

Поля:

- sesi50_cname - Містить покажчик на рядок містить ім'я комп'ютера встановив сесію.
- sesi50_username - Містить покажчик на рядок містить ім'я користувача встановив сесію.
- sesi50_key - містить значення за допомогою якого ми будемо завершувати сесію.
- sesi50_num_conns - містить число підключень зроблених протягом сесії.
- sesi50_num_opens - містить кількість файлів відкритих протягом сесії.
- sesi50_time - містить час у секундах протягом якого сесія була активна.
- sesi50_idle_time - містить час у секундах протягом якого сесія була неактивна.
- sesi50_protocol - містить ім'я протоколу, за допомогою якого клієнт зв'язується з сервером.
- Pad1 - невеликий вирівнювач структури, не використовується структура SESSION_INFO_502.

Таблиця 3.10

Структура SESSION_INFO_502

```
type
 TSessionInfo502 = packed record
   Sesi502_cname       : PWideChar;
   Sesi502_username    : PWideChar;
   Sesi502_num_opens   : DWORD;
   Sesi502_time        : DWORD;
   Sesi502_idle_time   : DWORD;
   Sesi502_user_flags  : DWORD;
   Sesi502_cltype_name : PWideChar;
   Sesi502_transport   : PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;
```

Код отримання поточних сесій знаходиться в дод. В.

Завершення сесій

- ServerName - повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо завершується сесія у себе то даному параметру потрібно присвоїти NIL.
- uncClientName - повинен містити ім'я клієнта чия сесія завершується, якщо параметр NIL, завершаться всі сесії зазначені в параметрі username.
- username - повинен містити ім'я користувача чия сесія завершується, якщо параметр NIL, завершаться всі сесії зазначені в параметрі uncClientName.

Таблиця 3.11

Завершення сесії

```
procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
```

Продовження таблиці 3.11

```
begin
  if not IsNT(OS) then Close;
  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index;
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, 'NetSessionDel');
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    CNameNT := PWChar(WideString('\\'+lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, 'NetSessionDel');
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
       CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i];
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;
```

Параметри визначення вхідного та вихідного трафіку наступні:

- pIfTable - повинен містити покажчик на структуру;
- pdwSize - повинен містити розмір структури;
- bOrder - вказує, чи потрібна сортування в повернутому масиві.

В якості першого параметра функція використовує покажчик на структуру, ось саме опис структури.

Таблиця 3.12

Визначення вхідного і вихідного трафіку

```
type
 TMibIfTable = packed record
   dwNumEntries: DWORD;
   Table       : TMibIfArray;
 end;
 PMibIfTable = ^ TMibIfTable;
type
  TMibIfRow = packed record
    wszName           : array[0..255] of WideChar;
    dwIndex           : DWORD;
    dwType            : DWORD;
    dwMtu             : DWORD;
    dwSpeed           : DWORD;
    dwPhysAddrLen     : DWORD;
    bPhysAddr         : array[0..7] of Byte;
    dwAdminStatus     : DWORD;
    dwOperStatus      : DWORD;
    dwLastChange      : DWORD;
    dwInOctets        : DWORD;
    dwInUcastPkts     : DWORD;
    dwInNUCastPkts    : DWORD;
    dwInDiscards      : DWORD;
    dwInErrors        : DWORD;
    dwInUnknownProtos : DWORD;
    dwOutOctets       : DWORD;
    dwOutUCastPkts    : DWORD;
    dwOutNUCastPkts   : DWORD;
    dwOutDiscards     : DWORD;
    dwOutErrors       : DWORD;
    dwOutQLen         : DWORD;
    dwDescrLen        : DWORD;
    bDescr            : array[0..255] of Char;
  end;
  TMibIfArray = array [0..512] of TMibIfRow;
  PMibIfRow = ^TMibIfRow;
  PmibIfArray = ^TmibIfArray;
```

Для виведення на екран результатів аналізу використовують наступну функцію, яка містить такі параметри. Це є завершальним етапом аналізу та дає можливість отримати результати роботи:

- wszName - Покажчик на рядок містить ім'я інтерфейсу;
- dwIndex - Визначає індекс інтерфейсу;
- dwType - Визначає тип інтерфейсу (див. MSDN);
- dwMtu - Визначає максимальну швидкість передачі;
- dwSpeed - Визначає поточну швидкість передавання даних в секунду;
- dwPhysAddrLen - Визначає довжину адреси міститься в bPhysAddr;
- bPhysAddr - Містить фізичну адресу інтерфейсу (якщо простіше то його, трохи видозмінений, MAC адреса);
- dwAdminStatus - Визначає активність інтерфейсу;
- dwOperStatus - Містить поточний статус інтерфейсу (див. MSDN);
- dwLastChange - Містить останній змінений статус;
- dwInOctets - Містить кількість байт прийнятих через інтерфейс;
- dwInUcastPkts - Містить кількість направлених пакетів прийнятих інтерфейсом;
- dwInNUCastPkts - Містить кількість ненапрямлених пакетів прийнятих інтерфейсом (включаючи Броадкаст і т.п.);
- dwInDiscards - Містить кількість забракованих вхідних пакетів (навіть якщо вони не містили помилки);
- dwInErrors - Містить кількість вхідних пакетів містять помилки;
- dwInUnknownProtos - Містить кількість забракованих вхідних пакетів зі структурою невідомого протоколу;
- dwOutOctets - Містить кількість байт відправлених інтерфейсом;
- dwOutUCastPkts - Містить кількість направлених пакетів відправлених інтерфейсом;
- dwOutNUCastPkts- Містить кількість ненапрямлених пакетів відправлених інтерфейсом (включаючи Броадкаст і т.п.);

- dwOutDiscards- Містить кількість забракованих вихідних пакетів (навіть якщо вони не містили помилки);
- dwOutErrors- Містить кількість вихідних пакетів містять помилки;
- dwOutQLen - Містить довжину черги даних;
- dwDescrLen - Містить розмір масиву bDescr;
- bDescr - Містить опис інтерфейсу.

Таблиця 3.12

Отримання результатів моніторингу мережевого трафіку

```
procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  type TMAC = array [0..7] of Byte;
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '00-00-00-00-00-00' else
    begin
      Result := '';
      for i:= 0 to Length -2 do
        Result := Result + IntToHex(Value[i],2)+'-';
      Result := Result + IntToHex(Value[Length-1],2);
    end;
  end;
var
  FLibHandle   : THandle;
  Table        : TMibIfTable;
  i            : Integer;
  Size         : Integer;
begin
  tmrTraffic.Enabled := False;
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;
  FLibHandle := LoadLibrary('IPHLPAPI.DLL');
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, 'GetIfTable');
  if not Assigned(GetIfTable) then
  begin
```

Продовження табл. 3.12

```
FreeLibrary(FLibHandle);
 Close;
  end;
  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, False ) = 0 then
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin
        Caption := String(Table.Table[i].bDescr);
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen));
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets));
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets));      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := True;
end;
```

3.2 Розробка графічного інтерфейсу

Створення дружнього інтерфейсу є пріорітетною задачею для даного програмного забезпеченя. Це допоможе у використанні програми користувачами з різним рівнем знань в області інформатики. В даному розділі показана візуалізація, зка була сторена за допомогою візуальних компонентів delphi 7. Мксимально спрощений інтерфейс дозволить отримувати лише ту інформацію, яка єкорисною для адмністратора мережі. Також, зі зменшенням навантаження на пристрій збільшиться продуктивність, що важливою складовою у випадку використання не надто потужного комп'ютеру. Перед даною стемою моніторингу мережевої активності стояла задача отримання максимально точних даних, для цього було збільено кількість фільтрів, які дозволять задавати більшу кількість параметрів та отримувати точніші результати без втрати продуктивності пристрою.

На Рис. 3.1 показані результати роботи табл. 3.1, табл. 3.2.

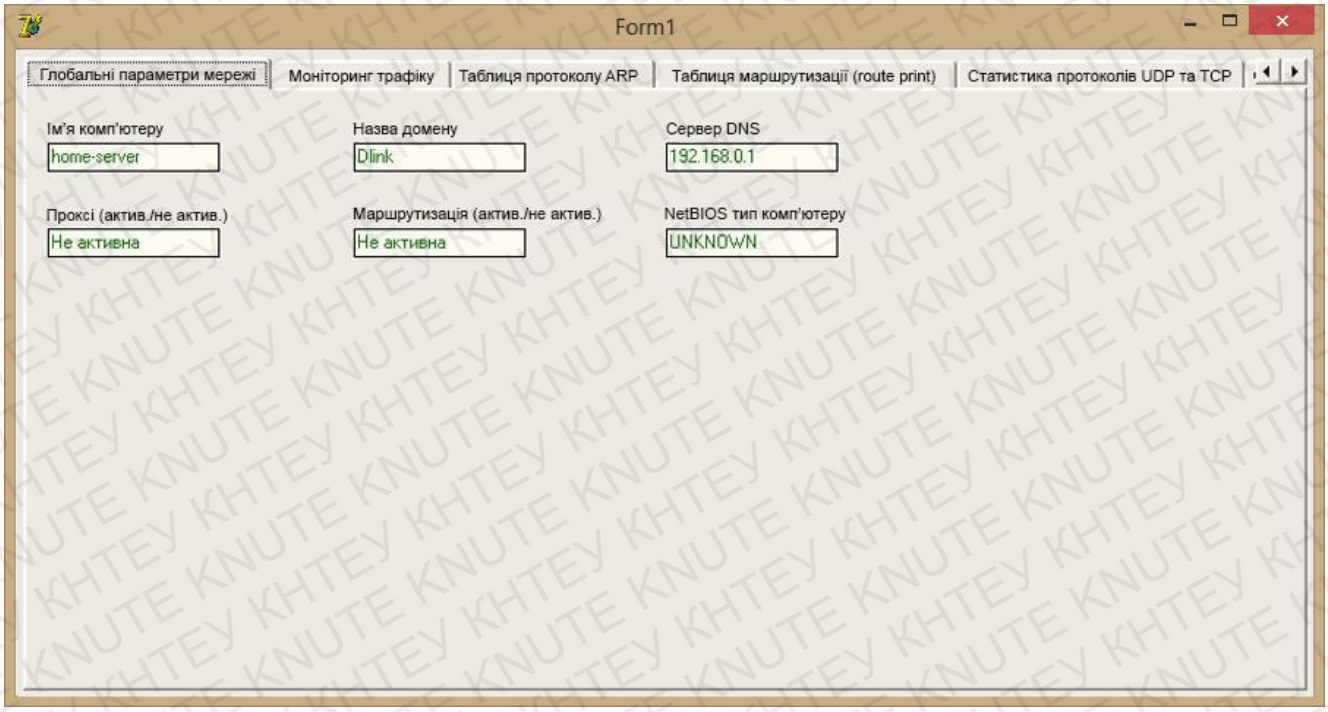


Рис 3.1 Глобальні параметри мережі в програмі

Розділ «Моніторинг трафіку» на рис. 3.2 є результатами роботи табл. 3.3, табл. 3.4, табл 3.5.

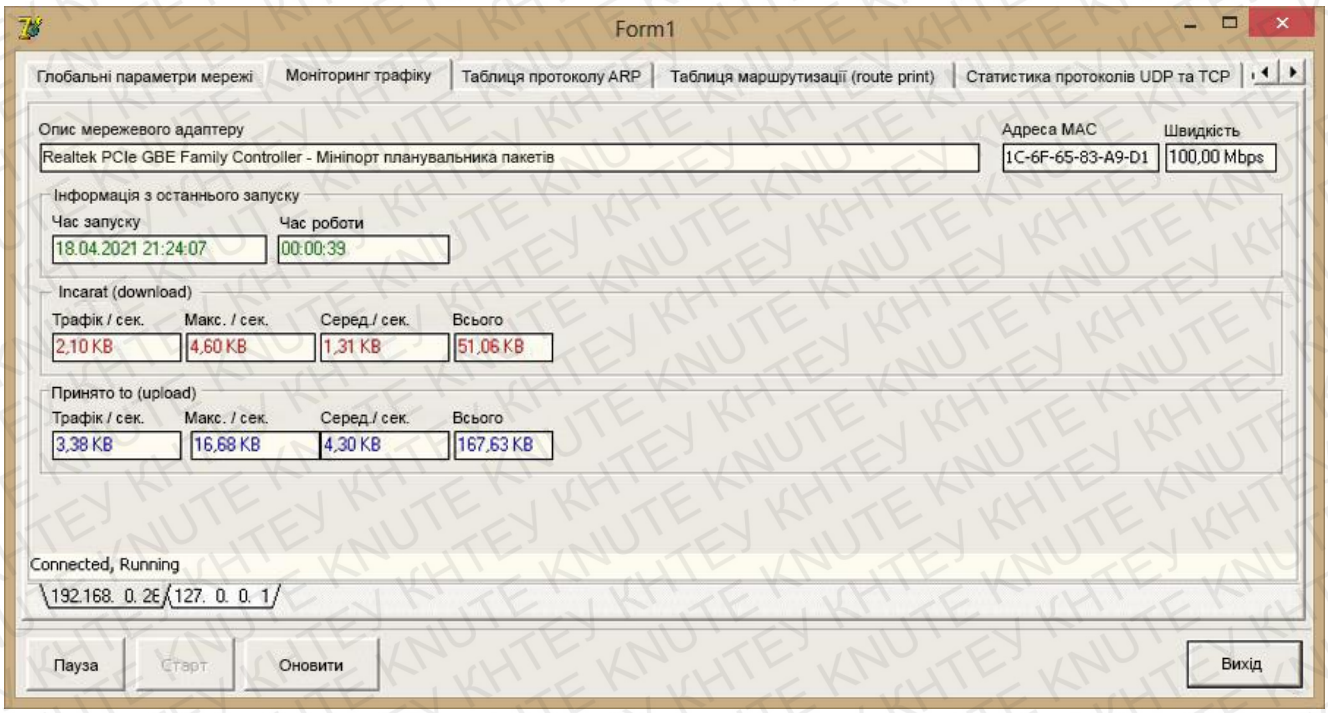
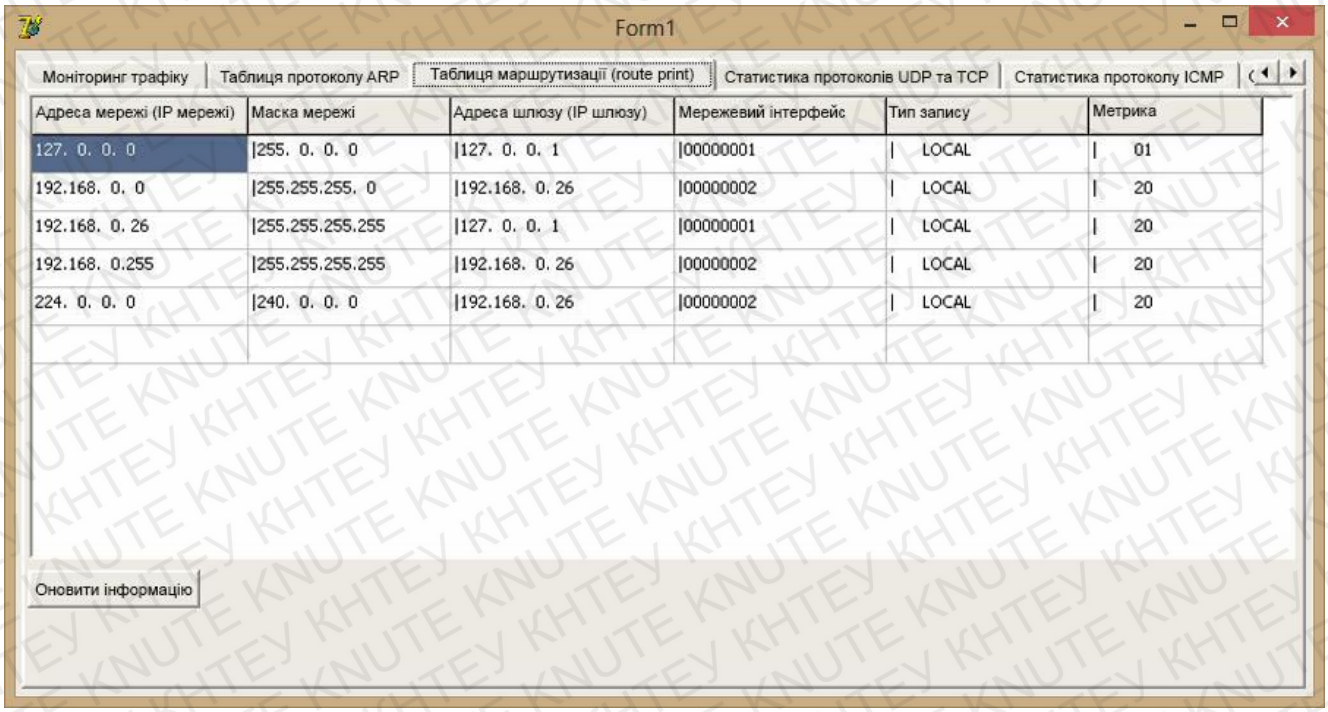


Рис 3.2 Відображення мережевої активності мережевих інтерфейсів

Розділ «Таблиця маршрутизації (route print)» на рис. 3.3 є результатами роботи табл. 3.6, табл. 3.7, табл 3.8.



Адреса мережі (IP мережі)	Маска мережі	Адреса шлюзу (IP шлюзу)	Мережевий інтерфейс	Тип запису	Метрика
127. 0. 0. 0	255. 0. 0. 0	127. 0. 0. 1	00000001	LOCAL	01
192.168. 0. 0	255.255.255. 0	192.168. 0. 26	00000002	LOCAL	20
192.168. 0. 26	255.255.255.255	127. 0. 0. 1	00000001	LOCAL	20
192.168. 0.255	255.255.255.255	192.168. 0. 26	00000002	LOCAL	20
224. 0. 0. 0	240. 0. 0. 0	192.168. 0. 26	00000002	LOCAL	20

Рис 3.3 Таблиця маршрутизації комп'ютера

Розділ «Статистика протоколів UDP та TCP» на рис. 3.4 є результатами роботи табл. 3.9, табл. 3.10.

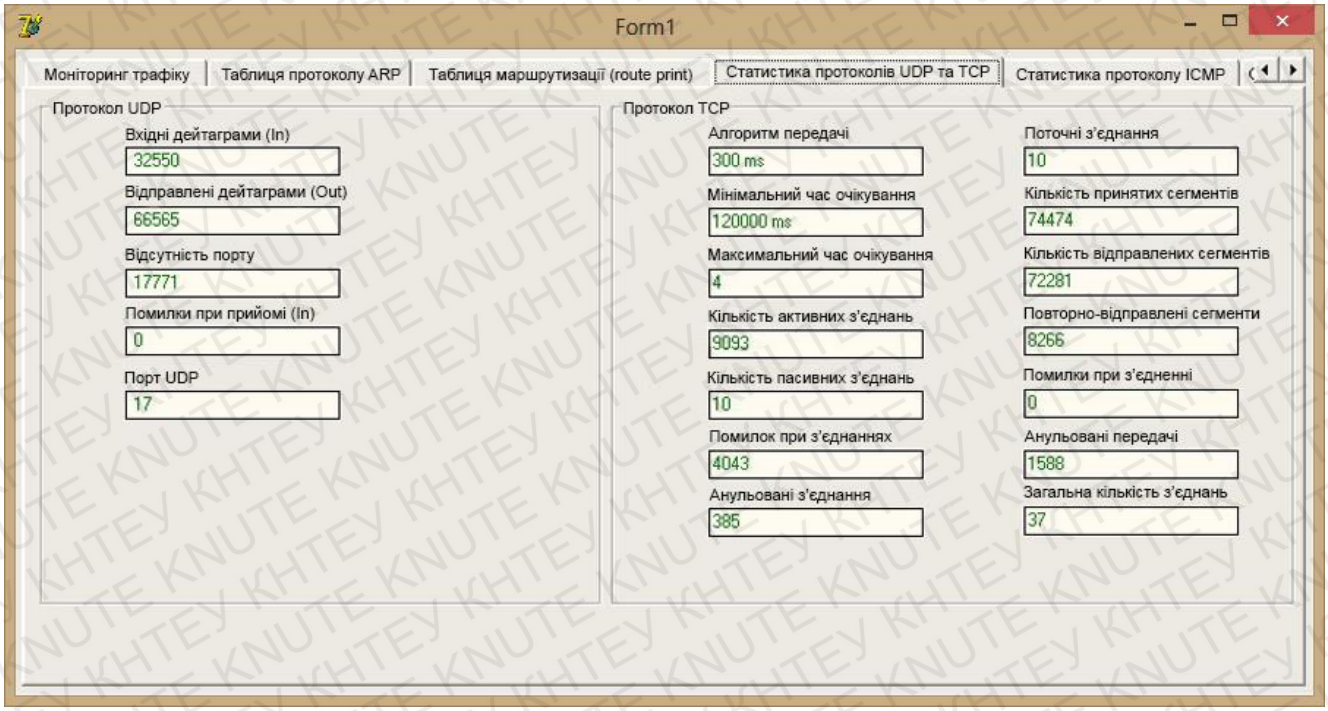


Рис 3.4 Параметри мережевої активності протоколів UDP і TCP

Розділ «Статистика протоколу ICMP» на рис. 3.5 є результатами роботи табл. 3.11, табл. 3.12.

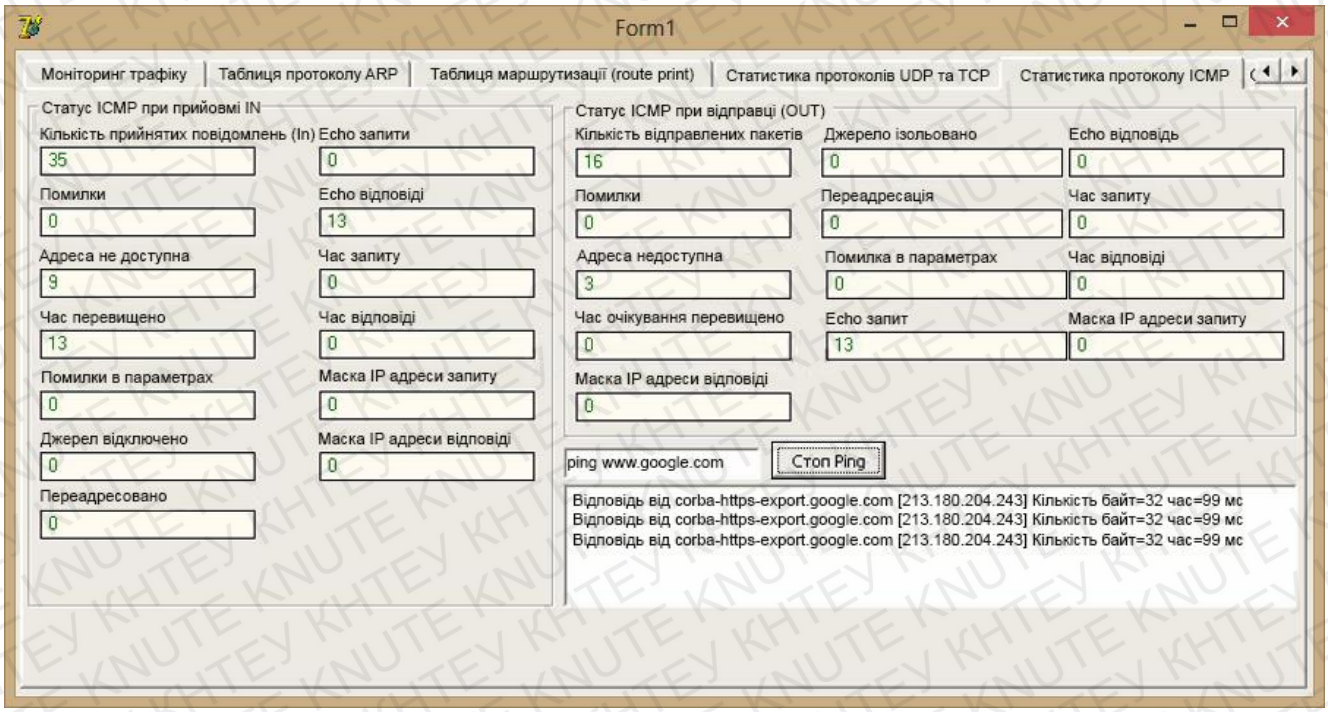


Рис 3.5 Відображення і аналіз процесу роботи команди ping протоколу ICMP

3.3 Особливості роботи та можливості масштабування програмного забезпечення

Важливим практичним питанням при використанні будь-якого засобу аналізу трафіку є питання масштабування при збільшенні ширини аналізованого каналу. Поширений спосіб вирішення - додавання в схему ще одного пристрою обробки трафіку. Однак при цьому також потрібно вирішувати задачу розподілу мережевих пакетів з вихідного каналу по пристроях обробки трафіку. Для вирішення цього завдання в разі широких вихідних каналів використовуються спеціальні мережеві пристрої – пакетні брокери або балансувальник, основним завданням яких є розподіл мережевих пакетів, отриманих з набору вхідних мережевих інтерфейсів по набору вихідних мережевих інтерфейсів. Залежно від логіки, яка застосовується при цьому виборі, можна виділити статичну і динамічне балансування.

При статичної балансуванню стандартними схемами комутації є:

- One-to-One - кожен вхідний порт відображається на окремий вихідний;
- Any-to-Any - будь-який вхідний порт на будь-який вихідний;

- Many-to-One - агрегація даних з декількох вхідних портів в один вихідний;
- One-to-Many - віддзеркалення трафіку на кілька споживачів;

При динамічному балансуванню, важливою функціональністю є можливість розбору заголовків різних протоколів низького рівня (таких як Ethernet, VLAN, MPLS та ін.). Це важливо насамперед для того щоб була можливість передавати пакети, що відносяться до одного потоку транспортного рівня на один мережевий інтерфейс, тобто до одного пристрою обробки трафіку. Зокрема, це дозволяє коректно виконувати TCP-нормалізацію.

Також це дозволяє зменшити ефект перестановки пакетів в рамках одного потоку, що, в свою чергу, підвищує ефективність алгоритмів IP дефрагментації і TCP-нормалізації. Наявність такого функціоналу говорить про тому, що самі балансувальник є апаратним засобом мережевого аналізу. Крім того, багато хто з них також підтримують апаратну фільтрацію по вмісту пакетів, що говорить про те, що балансувальник потенційно відносяться до DPI рішенням.

Висновки розділу

Створено оптимальну модель функціонування програмного забезпечення, розроблено програмний код та створено інтерфейс користувача.

ВИСНОВКИ

В процесі виконання випускної кваліфікаційної роботи було зроблено наступне:

1. Проведено аналіз та класифікацію систем моніторингу, що дало змогу обрати найефективніші параметри для майбутнього програмного забезпечення.
2. Провівши аналіз технологій було визначено, за допомогою яких технологій найзручніше реалізувати дане програмне забезпечення.
3. Дослідження мов програмування дозволило порівняти існуючі та обрати оптимальне. За оптимальне програмне забезпечення було обрано Delphi 7.
4. Структуровано інформацію та створено оптимальну модель функціонування програмного забезпечення.
5. Розроблено програмний код що дозволяє використовувати програмне забезпечення на слабких комп’ютерах.
6. Створено інтерфейс користувача.
7. За результатами розробки отримано програмного забезпечення для моніторингу мережевої активності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. L.Zhanh and J.Tang, Characterization and performance study of IP traffic in WDM networks // Computer communications, 2017., No.24, pp.1702-1713.
2. Jan Engelhardt and Nicolas Bouliane. Writing Netfilter modules. Revised, February 07, 2011. 3. Ed Wilson, “Network Monitoring and Analysis: A Protocol Approach to Troubleshooting”, Prentice Hall PTR, 2015.
3. Laura Chappell. "Wireshark (R) 101: Essential Skills for Network Analysis". Laura Chappell University, 2015.
4. Shui Yu, “Distributed Denial of Service Attack and Defense” (SpringerBriefs in Computer Science), Springer–Verlag New York, 2016.
5. Lysenko S. Botnet detection technique for corporate area network / Lysenko S., Savenko O., Kryshchuk A., Kljots Y. Proceedings of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), Berlin, DE, IEEE. – 2018. – P. 315-320.
6. Pomorova O. Multi-agent based approach for botnet detectionin a corporate area network using fuzzy logic / Pomorova O., Savenko O., Lysenko S., Kryshchuk A. // Computer Networks 20th International Conference, CN 2017, Lwówek Śląski, Poland, June 17–21. – 2017. – P. 243–254
7. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker. P4: Programming protocol-independent packet processors. SIGCOMM Computer Communications Review, 2019.
8. A. Bremler-Barr, Y. Harchol, D. Hay, Y. Koral. Deep packet inspection as a service. In CoNEXT, pages 271–282, 2019.
9. Cascarano N, Ciminiera L, Risso F. Optimizing deep packet inspection for high-speed traffic analysis. Network System Manager. 2019; 19(1):7–31
10. Duffield N., Lund C. “Predicting Resource Usage and Estimation Accuracy in an IP Flow Measurement Collection Infrastructure”. ACM Internet Measurement Conference, 2020

11. Duffield, N.; Lund, C.; Thorup, M., “Learn more, sample less: control of volume and variance in network measurement”, IEEE Transactions in Information Theory, vol. 51, no. 5, pp. 1756-1775, 2019.
12. R. Sommer and A. Feldmann. NetFlow: Information loss or win? In ACM SIGCOMM Internet Meas. Workshop, 2019.
13. Cascarano N, Ciminiera L, Risso F. Optimizing deep packet inspection for high-speed traffic analysis. Network System Manager. 2028; 19(1):7–31.
14. Koloud Al-Khamaiseh, Shadi ALShagarin. A Survey of String Matching Algorithms. Int. Journal of Engineering Research and Applications. ISSN : 2248-9622, Vol. 4, Issue 7(Version 2), July 2018, pp.144-156
15. T. Farah, and L. Trajkovic, "Anonym: A tool for anonymization of the Internet traffic." In IEEE 2018 International Conference on Cybernetics (CYBCONF), 2018, pp. 261-266.S. Kumar and P. Crowley. Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection. In Proc. of SIGCOMM, 2018.
16. A. Feitoza Santos, S. F. de Lacerda Fernandes, P. Gomes Lopes Junior, D. Fawzi Hadj Sadok, and G. Szabo, “Multi-gigabit traffic identification on gpu,” in Proceedings of the First Edition Workshop on High Performance and Programmable Networking, ser. HPPN ’13. New York, NY, USA: ACM, 2018, pp. 39–44.
17. Callado A., Kamienski C., Szabo G., Gero B., Kelner J., Fernandes S., Sadok D. A Survey on Internet Traffic Identification; Communicatio 2, New York, NY, USA: ACM, 2017, pp. 156–157.
18. Alisha Cecil, “A summary of Network Traffic Monitoring and Analysis Techniques Whitepaper”, 2016, [Електронний ресурс]. – Режим доступу: http://www.cs.wustl.edu/~jain/cse567- 06/ftp/net_monitoring/index.html
19. Tee Huu, “Evaluation of a Multi-Agent System for Simulation and Analysis of Distributed Denial-of-Service Attacks. New edition”, 2020, [Електронний ресурс]. – Режим доступу: http://www.dtic.mil/dtic/tr/fulltext/u2/a420448.pdf

20. Juniper Official Website, [Електронний ресурс]. – Режим доступу: http://www.juniper.net/techpubs/software/junoses/junos-es93/junos-es-swconfig-security/understanding-icmp-flood-attacks.html.
21. Моніторинг мережі [Електронний ресурс] // Режим доступу до матеріалу: http://www.4stud.info/networking/work2.html
22. Огляд методів аналізу і моніторингу мережевого трафіку [Електронний ресурс] // Режим доступу до матеріалу: http://it-bloknot.ru/?q=content/обзорметодов-анализа-и-мониторинга-сетевого-трафика.
23. Utilizing GPUDirect 3RD Party DMA Features for 10GbE NIC and GPU applications. [Електронний ресурс]. – Режим доступу: http://on-demand.gputechconf.com/gtc/2013/presentations/S3300-GPUDirect-DMAFeatures.pdf
24. DirectGMA. [Електронний ресурс]. – Режим доступу: http://developer.amd.com/tools-and-sdks/graphics development/fireprosdk/firepro-directgma-sdk/
25. IANA Service Name and Transport Protocol Port Number Registry. [Електронний ресурс]. – Режим доступу: http://www.iana.org/assignments/service-names-port-numbers/service-names-portnumbers.xhtml

Закриття локального ресурсу

```
procedure TMainForm.btnCloseSharesClick(Sender: TObject);
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close;
  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count -1 do
    if lbxShares.Selected[i] then Break;
  if not lbxShares.Selected[i] then Exit;
  ShareName := lbxShares.Items.Strings[i];
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,'NetShareDel');
    if not Assigned(NetShareDelNT) then //Прверка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);
    StringToWideChar(ShareName,NameNT,i);
    NetShareDelNT(nil,NameNT,0);
    FreeMem(NameNT);
  end else begin
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,'NetShareDel');
    if not Assigned(NetShareDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0);
    move(ShareName[1],Name9x[0],Length(ShareName));
```

```
    NetShareDel(nil,@Name9x,0);
  end;
  FreeLibrary(FLibHandle);
end;
```

ДОДАТОК Б

Відкриття локального ресурсу

```
procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL = 258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory;
  TmpName := InputBox('Share name','Enter name','Test');
  if TmpDir = '' then Exit;
  if not IsNT(OS) then Close;
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,'NetShareAdd');
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256;
    GetMem(TmpNameNT, TmpLength);
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT;
    ShareNT.shi2_type := STYPE_DISKTREE;
    ShareNT.shi2_remark := '';
    ShareNT.shi2_permissions := ACCESS_READ;
    ShareNT.shi2_max_uses := DWORD(-1);.
    ShareNT.shi2_current_uses := 0;.
    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT;
    ShareNT.shi2_passwd := nil;
```

```
    NetShareAddNT(nil,2,@ShareNT,nil);
    FreeMem (TmpNameNT);
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,'NetShareAdd');
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName));
    Share9x.shi50_type := STYPE_DISKTREE;
    Share9x.shi50_flags := SHI50F_RDONLY;
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0);
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir);
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0);
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0);
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;
```

ДОДАТОК В

Отримання поточної сесії

```
procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvSessions.Items.Clear;
  if not IsNT(OS) then Close;
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, 'NetSessionEnum');
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname);
        SubItems.Add(SessionInfo502^[i].sesi502_username);
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time));
          SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
      end;
    end;
  end else begin
   FLibHandle := LoadLibrary('SVRAPI.DLL');
   if FLibHandle <> 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, 'NetSessionEnum');
    if not Assigned(NetSessionEnum) then
```

```
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname);
        SubItems.Add(SessionInfo50[i].Sesi50_username);
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens));
SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time));
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key;      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;
```