ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	– база даних
ЛОМ	– локальна обчислювальна мережа
ОП	– оперативна пам'ять
ОС	– операційна система
ПЗ	– програмне забезпечення
ПК	– персональний комп'ютер
ПП	– програмний продукт
РС	– робоча станція
ТЗ	– технічне завдання
IP	– Internet Protocol
LAN	– Local Area Network
NAT	– Network Address Translation

ВСТУП

Бурхливий розвіток комп’ютерних мереж забезпечив можливості створення складних обчислювальних систем обробки даних різного призначення, орієнтованих на роботу в мережі Internet. Користувачі використовують глобальні мережі в повсякденній роботі, об'єднуючи віддалені офіси та використовуючі хмарні сховища. Зростання рівня автоматизації процесів обробки, зберігання і передачі інформації сприяє виникненню проблем з забезпечення безпеки від шкідливої діяльності зловмисників.

Необмежений і безконтрольний доступ співробітників організації в глобальну мережу є небезпечним для корпоративної мережі, тому виникає необхідність стежити, який співробітник скільки трафіку використовує. Аналіз трафіку дозволяє визначити тип інформації і проконтролювати її співробітникам внутрішньої безпеки організації. Облік трафіку часто потрібен і для діагностики мережі: виявлення «вузьких» місць та визначення реальної швидкісті передачі даних між хостами. Неконтрольований доступ сприяє перевантаженню корпоративної локальної мережі, тому потребує також обмеження мережевого трафіку.

Таким чином, інформаційна інфраструктура сучасного підприємства, яка інтегрує різнорідні мережі і системи, потребує моніторингу мережі, обліку трафіку і розмежування доступу в глобальну мережу. Слід відзначити, що способи віддаленого шкідливого впливу постійно вдосконалюються, а існуючі системи захисту не дозволяють своєчасно реагувати на зміни в цій сфері. На жаль, стандартні засоби MS Windows не володіють необхідною функціональністю для вирішення подібного роду завдань, тому актуальним є розробка спеціалізованого програмного забезпечення виявлення несанкціонованого трафіку і розробки актуальних засобів захисту інформації.

У зв'язку з цим метою дипломної роботи є розробка прототипу системи збору мережевого трафіку для аналізу, обліку та розмежування доступа у мережу.

В рамках роботи будуть вирішені наступні завдання:

– дослідження проблеми забезпечення безпеки мережі і порівняльний аналіз сучасних систем аналізу мережевого трафіку;
– обґрунтування вибору інструментальних засобів розробки програмної системи;
– розробка архітектури системи та моделювання стенду, необхідного для розробки та налагодження програмного забезпечення (ПЗ);
– проектування програмної системи;
– реалізація прототипу програмної системи збору, аналізу та обліку мережевого трафіку на основі розробленої архітектури і алгоритмів;
– тестування програмного продукту.

1 ДОСЛІДЖЕННЯ ІСНУЮЧИХ СИСТЕМ МОНІТОРИНГУ ТА АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

1.1 Поняття контроль і моніторинг локальних мереж

Для підтримки локальної обчислювальної мережі підприємства в постійному працездатному стані дуже необхідним є її безперервний контроль з боку мережевого адміністратора. Контроль – це головний і дуже важливий етап процедури керування мережею. Ця функція часто реалізується спеціалізованими програмними засобами. За допомогою засобів контролю мережевий адміністратор може виявити проблеми, які має ЛОМ та некоректні її налаштування, а потім виконати вірні налаштування та реконфігурацію вручну. Розглянемо два етапи процесу контролю мережі – моніторинг і аналіз [1][1)].

Етап аналізу являє собою більш інтелектуальний процес осмислення зібраної на етапі моніторингу інформації, порівняння її з даними, отриманими раніше, і визначення можливих причин сповільненої або ненадійної роботи мережі.

[1)] [1] Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы. СПб.: Питер, 2010. 882 с.

Етап моніторингу передбачає збір первинних даних про роботу мережі: статистику про кількість циркулюючих в мережі пакетів, обсягу даних, кількості мережевих адрес тощо [2][1)].

Завдання моніторингу зазвичай можуть вирішуватися:

- програмними і апаратними засобами;
- мережевими аналізаторами;
- вбудованими засобами моніторингу комунікаційних пристроїв;
- спеціалізованими програмними продуктами.

Мережеві монітори збирають дані про статистичні показники трафіку, наприклад середньої інтенсивності загального трафіку мережі, середньої інтенсивності потоку пакетів певного типу.

1.2 Класифікація засобів моніторингу та аналізу

Системи моніторингу мережі – це програмне забезпечення, що дозволяє відстежувати стан мережевих пристроїв, їх працездатність, справність і інші характеристики. При цьому системи моніторингу мережі дозволяють повідомляти адміністратору про виникнення будь-яких збоїв за допомогою відправки СМС-повідомлень, повідомлень на електронну пошту тощо. Системи моніторингу мережі можна розділити на ті, що відстежують продуктивність мережі і сигналізують при перевантаженні каналів, а також на ті, що виробляють моніторинг мережі з метою пошуку збоїв і інших проблем, пов'язаних з працездатністю серверного обладнання та інших систем.

Розглянемо групи інструменти, які зазвичай використовують для моніторингу та аналізу локальних мереж [3][2)]:

[1)] [2] Уилсон Э. Мониторинг и анализ сетей. Методы выявления неисправностей. М.: Лори, 2002. 288 с.

[2)] [3] Олифер Н. А. Средства анализа и оптимизации локальных сетей. URL: http://citforum.ru/nets/optimize/locnop_07.shtml (дата звернення 13.05.2019)

Системи керування мережею (Network Management Systems) – централізовані програмні системи, які збирають дані про стан мережевих пристроїв і інформацію про трафік. Функціонал даних програм не обмежений моніторингом та аналізом мережі, вони також можуть здійснювати керування мережею: налаштування та зміна адресних таблиць комутаторів та іншого обладнання, включення і відключення портів пристроїв, тощо.

Вбудовані системи діагностики і керування (Embedded systems). Виконуються у вигляді програмно-апаратних модулів, які встановлюються в комунікаційне обладнання, або в операційну систему. Вони дозволяють керувати і діагностувати тільки той пристрій, на якому знаходяться. Зазвичай, вбудовані модулі керуванняя також виконують роль SNMP-агентів, передаючи дані про стан пристрою в систему керування.

Засоби керування системою (System Management). Виконують функції, аналогічні функціям систем керування, але по відношенню до інших об'єктів. У першому випадку об'єктом керування є програмне і апаратне забезпечення комп'ютерів мережі, а у другому – комунікаційне обладнання. При цьому частина функцій цих двох видів систем можуть дублюватися (наприклад, засоби керування системою можуть проводити найпростіший аналіз трафіку).

Аналізатори протоколів (Protocol analyzers) – це програмні або апаратно-програмні системи, які використовуються тільки для моніторингу і аналізу трафіку в мережах. Хорошим аналізатором вважається той, який може захоплювати і декодувати пакети великої кількості протоколів, що застосовуються у мережах – приблизно декількох десятків. Ця група систем може встановлювати деякі логічні умови для захоплення окремих пакетів і виконувати повне декодування пакетів, тобто відображати в зручній для користувача формі вкладеність пакетів протоколів різних рівнів з розшифровкою змісту кожного з полів пакету [4][1)].

[1)] [4] Лавров А. А., Лисс А. Р., Яновский В. В. Мониторинг и администрирование в корпоративных вычислительных сетях. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 60 с.

Устаткування для діагностики і сертифікації кабельних систем. З назви стає ясно призначення цієї групи. умовно можна виділити чотири підтипи подібного обладнання: кабельні сканери, мережеві монітори, мультиметри і прилади для сертифікації кабельних систем.

Експертні системи з'єднують людські знання про виявлення причин аномальної роботи мереж і можливі способи повернення мережі в робочий стан. Найчастіше вони представлені у вигляді окремих підсистем інших засобів моніторингу та аналізу мереж, розглянутих раніше. До простого варіанту експертної системи відноситься контекстно залежна help-система, а більш складні являють собою так звані бази знань, які мають елементи штучного інтелекту.

Багатофункціональні пристрої аналізу і діагностики. Це недорогі портативні прилади з функціоналом декількох пристроїв: кабельних сканерів, програм мережевого управління і аналізаторів протоколів.

Також варто відзначити ще два способи моніторингу мережі. Перший – це маршрутизатор-орієнтований. Він являє собою моніторинг, вбудований безпосередньо в маршрутизатор і не вимагає додаткової установки іншого забезпечення. Другий спосіб, відповідно, – це не орієнтований на маршрутизатори, тобто це підібране самим фахівцем необхідне апаратне і програмне забезпечення для поточних потреб.

Використання систем моніторингу мережі дозволяє виконувати не тільки технічні завдання, описані вище, але і бізнес-завдання, такі як обгрунтування витрат на модернізацію обладнання або інфраструктури, що дозволило б знизити кількість проблемних вузлів. За рахунок підтримки мережі в працездатному стані і своєчасному усуненні її проблем компанія може знизити свої витрати на відновлення і повторюється усунення неполадок. А також підвищити свою репутацію в частині надійності надання власних сервісів клієнтам (це особливо актуально в разі, якщо організація надає доступ до веб-сайту, здійснює свою діяльність через інтернет або надає свої ресурси користувачам в якості хмарного сервісу). У той же час коли керівництво мо-

же стежити за якістю надходять послуг (наприклад, канал інтернет провайдера), при виникненні будь-яких збоїв з вини постачальника організація-споживач має право вимагати компенсацію, якщо такі збої відображені в угоді про якість обслуговування.

1.3 Порівняльний аналіз існуючих систем моніторингу локальних мереж

Розглянемо сучасні програми для моніторингу локальної мережі, які є актуальними в 2019 році.

Програма Total Network Monitor 2 (TNM 2) – розробка російської компанії Softinventive Lab (рис.1.1). Доступне і ефективне програмне рішення для мережевого моніторингу діяльності серверних машин. Одним з основних програмованих компонентів Total Network Monitor 2 є монітори, які і виконують чисельні перевірки з необхідною періодичністю. Вони дозволяють відстежити практично будь-який параметр, починаючи від доступності серверів в мережі і закінчуючи перевіркою стану сервісів. Здатні самостійно усувати первинні наслідки неполадок – тобто те, що спочатку системний адміністратор виконував вручну. У звітності зберігається вся інформація, пов'язана з кожною перевіркою, яка була проведена обраним монітором. Вартість за 1 копію цього додатка становить близько 2 500 грн [5][1)].

Застосунок Observium (рис.1.2), робота якого заснована на використанні протоколу SNMP, дозволяє не тільки дослідити стан мережі будь-якого масштабу в режимі реального часу, а й аналізувати рівень її продуктивності [6][2)]. Це рішення інтегрується з обладнанням від Cisco, Windows, Linux, HP, Juniper, Dell, FreeBSD, Brocade, Netscaler, NetApp і інших вендорів.

[1)] [5] Total Network Monitor 2. Офіційний сайт. URL: https://www.total-network-monitor.ru/ (дата звернення 13.05.2019)

[2)] [6] Observium. Офіційний сайт. URL: http://www.observium.org/ (дата звернення 13.05.2019)

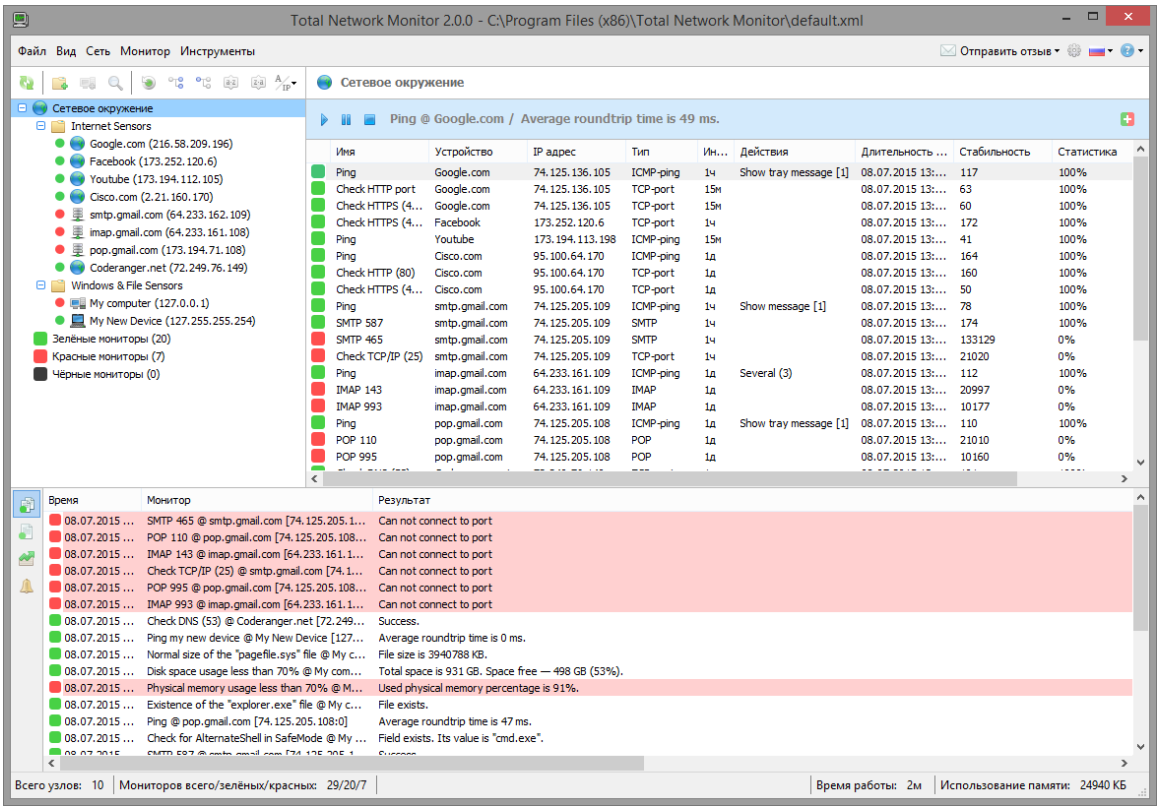

Рисунок 1.1 – Головне вікно програми Total Network Monitor 2

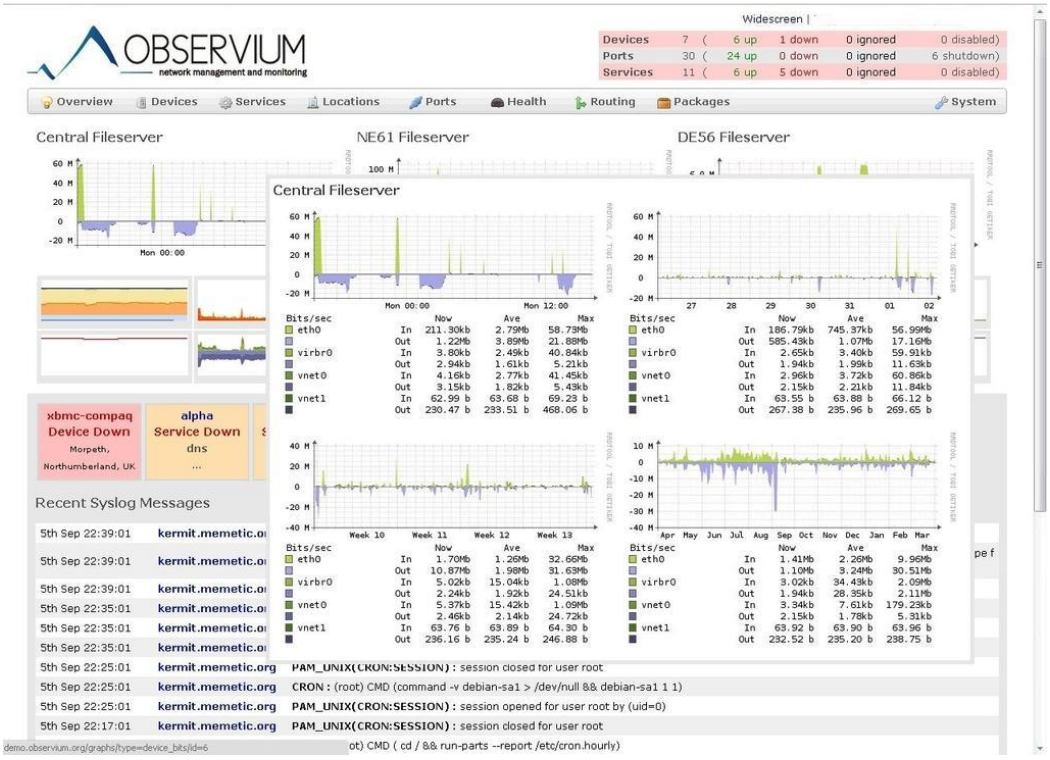

Рисунок 1.2 – Головне вікно програми Observium

Завдяки ідеально відпрацьованому графічному інтерфейсу, ці програми надає системним адміністраторам масу варіантів для налаштувань – починаючи від діапазонів для автовизначення і закінчуючи даними протоколу SNMP, необхідними для збору інформації про мережу. Також вони отримують доступ до даних про технічні характеристики всього обладнання, яке зараз підключено до мережі. Всі звіти, які формуються за допомогою аналізу журналу подій, Observium може представляти у вигляді діаграм і графіків, наочно демонструючи "слабкі" сторони мережі. Є можливість використовувати як демо-версію (яка має недостатній набор можливостей), так і платну ліцензію, річна вартість використання якої становить 200 фунтів стерлінгів.

Програма Nagios [7][1)] – це просунуте рішення для моніторингу, керування, яке засновано на веб-інтерфейсі (рис.1.3). Він складний в освоєнні, проте, завдяки інтернет-співдружності і добре відпрацьованої документації, може бути освоєний за кілька тижнів.

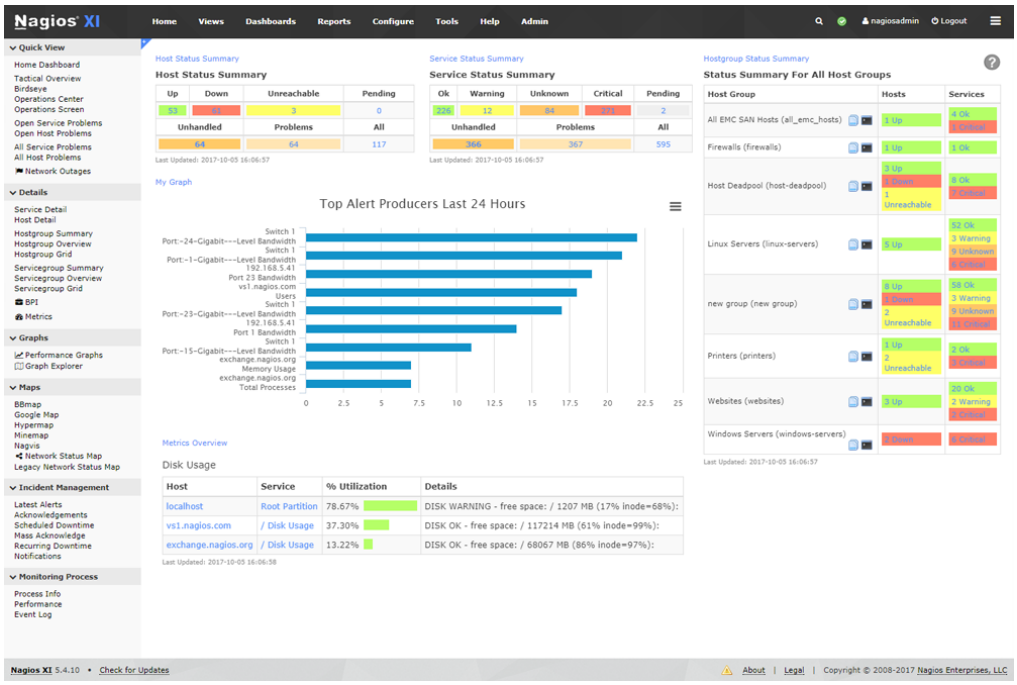

Рисунок 1.3 – Головне вікно програми Nagios

1) [7] Nagios. Офіційний сайт. URL: https://www.nagios.org/ (дата звернення 13.05.2019)

За допомогою Nagios системні адміністратори отримують можливість віддалено регулювати обсяг навантаження обладнання користувача або комутатори, маршрутизатори, сервери, стежити за ступенем завантаженості резервів пам'яті в базах даних, стежити за фізичними показниками частин мережевого обладнання (наприклад, температурою материнської плати , згоряння якої є однією з найбільш частих поломок в даній сфері) та ін. Що стосується виявлення мережевих аномалій, Nagios автоматично відправляє тривожні повідомлення на попередньо встановлений адміністратором адресу, будь то адреса електронної пошти або номер телефону мобільного оператора. Має 60-ти днівну доступну безкоштовнау демо-версію.

Програмний компонент PRTG Network Monitor [8][1)] сумісний з пристроями на базі ОС Windows, призначений для моніторингу мереж (рис.1.4). Він не безкоштовний (безкоштовним є лише пробний 30-ти денний період), використовується також для виявленні мережевих атак. Серед мережевих сервісів PRTG: інспекція пакетів, аналіз і збереження статистичних даних в БД, перегляд карти мережі в режимі реального часу, збір технічних параметрів про пристрої, підключених до мережі, а також аналіз рівня навантаження на мережеве обладнання.

Програма Kismet [9][2)] – це корисний open-source застосунок для системних адміністраторів, яке дозволяє всебічно аналізувати мережевий трафік, виявляти в ньому аномалії, запобігати збоїв і може бути використано з системами на базі * NIX / Windows / Cygwin / macOS (рис.1.5). Kismet нерідко використовується саме для аналізу бездротових локальних мереж на основі стандарту 802.11 b (в тому числі, навіть мереж з прихованим SSID). З його допомогою можна знайти точки доступу, що некоректно сконфігуровані і навіть нелегально працюють (які зловмисники використовують для перехоп-

[1)] [8] PRTG Network Monitor. Офіційний сайт. URL: https://www.ru.paessler.com/ (дата звернення 13.05.2019)

[2)] [9] Kismet. Офіційний сайт. URL: https://www.kismetwireless.net/ (дата звернення 13.05.2019)

лення трафіку) та інші приховані пристрої, які можуть бути потенційно "шкідливі" для мережі. Для цих цілей в застосунку дуже добре опрацьована можливість виявлення різних типів мережевих атак – як на рівні мережі, так і на рівні каналів зв'язку. Як тільки одна або кілька атак будуть виявлені, системний адміністратор отримає тривожний сигнал і зможе вжити заходів щодо усунення загрози.

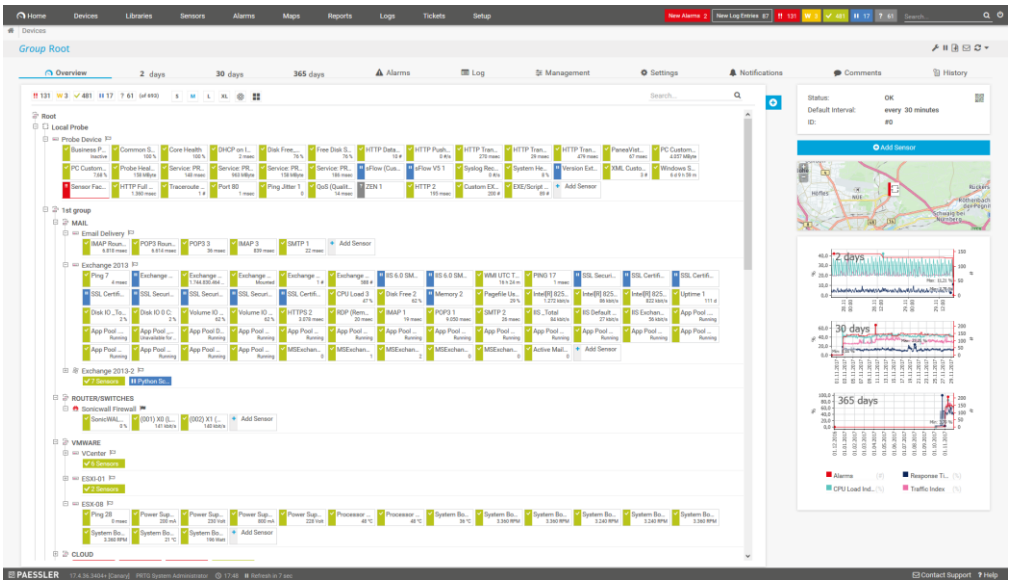

Рисунок 1.4 – Головне вікно програми PRTG Network Monitor

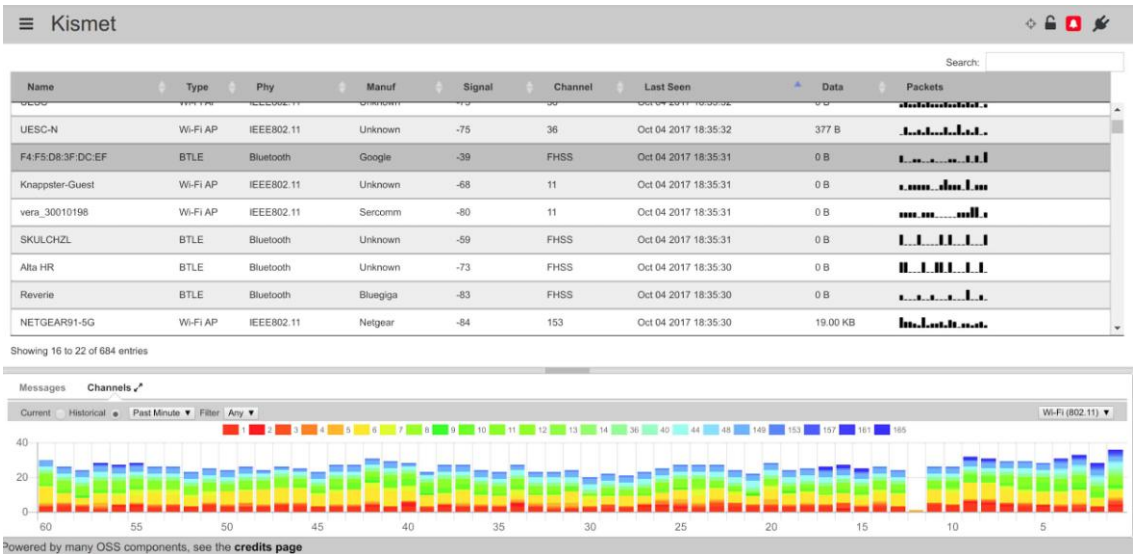

Рисунок 1.5 – Головне вікно програми Kismet

Програма WireShark [10][1)] – безкоштовний open-source аналізатор трафіку, що надає своїм користувачам просунутий функціонал і визнаний зразковим рішенням в області мережевої діагностики (рис.1.6). Він ідеально інтегрується з системами на базі * NIX / Windows / macOS. Замість не дуже добре зрозумілих для новачків веб-інтерфейсів і CLI, в яких потрібно вводити запити спеціальною програмною мовою, дане рішення використовує GUI. Розгорнувши і налаштувавши його одного разу на своєму сервері, можна отримати централізований елемент для моніторингу за найдрібнішими змінами в роботі мережі і мережевих протоколах. Таким чином, є можливість на ранніх етапах виявляти і ідентифікувати проблеми, що виникають в мережі.

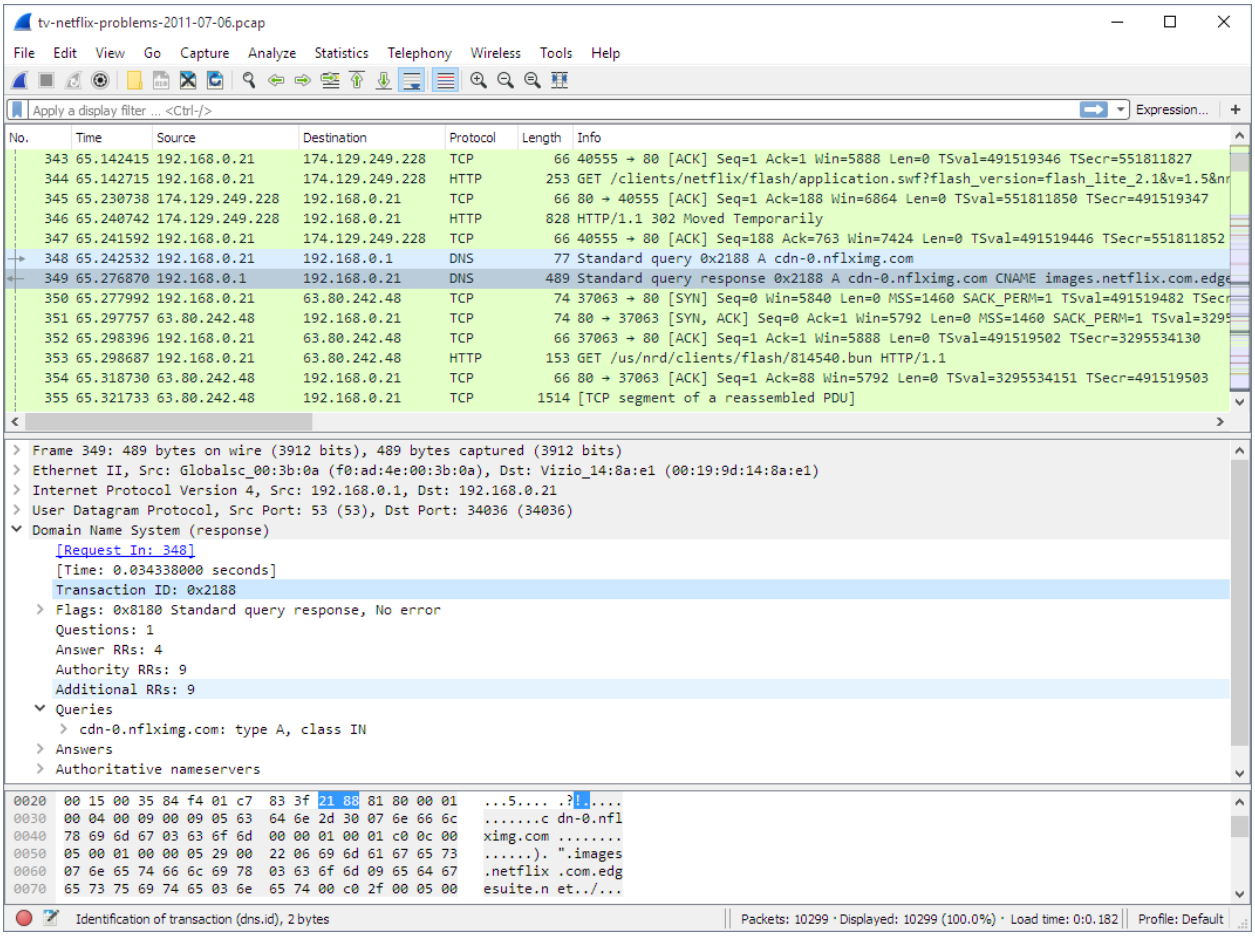

Рисунок 1.6 – Головне вікно програми WireShark

[1)] [10] Wireshark. Офіційний сайт. URL: https://www.wireshark.org/ (дата звернення 13.05.2019)

Програма NeDi [11][1)] – це повністю безкоштовне ПЗ, яке сканує мережу по MAC-адресам (також серед допустимих критеріїв пошуку є IP-адреси і DNS) і складає з них власну БД (рис.1.7). Для роботи цей програмний продукт використовує веб-інтерфейс. Таким чином, можна в режимі он-лайн спостерігати за всіма фізичними пристроями і їх розташування в рамках локальної мережі. Деякі професіонали задіють NeDi для пошуку пристроїв, які використовуються нелегально (наприклад, вкрадені). Для підключення до комутаторів або маршрутизаторів ці програми використовує протоколи CDP / LLDP. Це дуже корисне, хоча і непросте в освоєнні рішення.

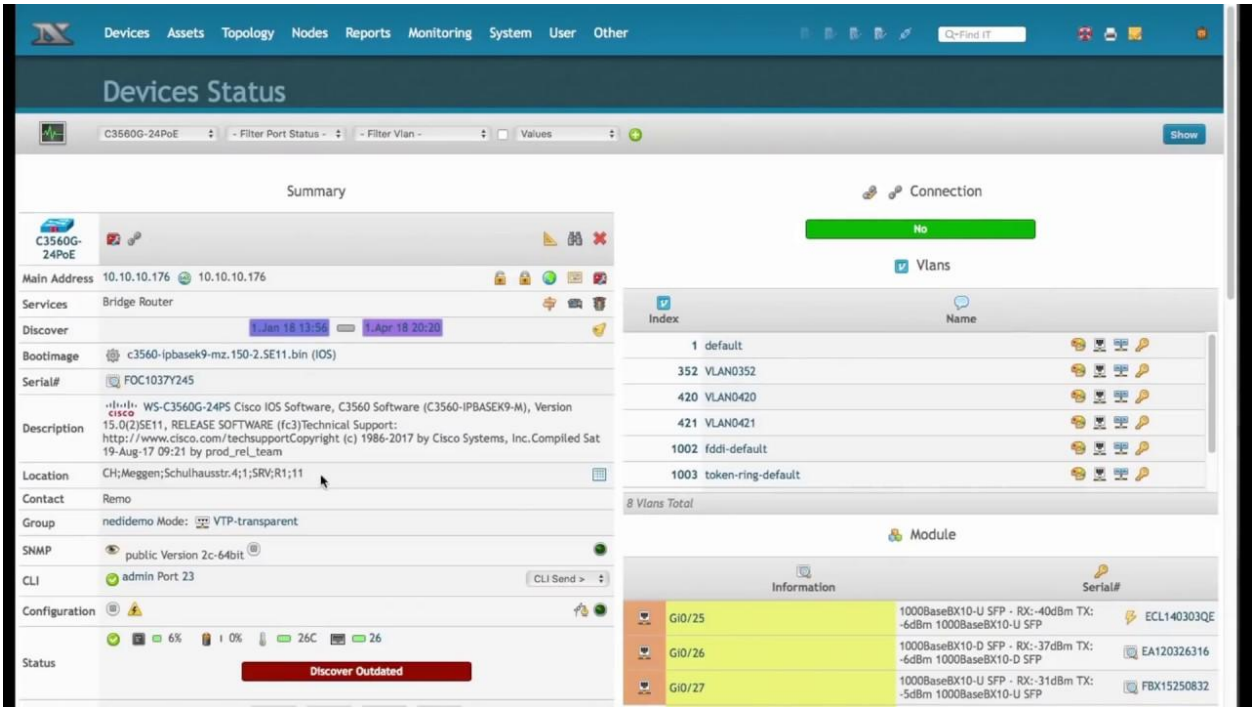

Рисунок 1.7 – Головне вікно програми NeDi

Система моніторингу Zabbix [12][2)] – це універсальне рішення для мережевого моніторингу з відкритим вихідним кодом, яке може бути налаштоване під окремі мережеві моделі (рис.1.8.). В основному, воно призначене для

[1)] [11] Nedi. Офіційний сайт. URL: https://www.nedi.ch/ (дата звернення 13.05.2019)

[2)] [12] Zabbix. Офіційний сайт. URL: https://www.zabbix.com/ (дата звернення 13.05.2019)

систем, які володіють багатосерверною архітектурою (зокрема, Zabbix інтегрується з серверами Linux / FreeBSD / Windows). Цей за стосунок дозволяє одночасно управляти сотнями мережевих вузлів, що робить його вкрай ефективним інструментом в організації роботи сисадмінів, які працюють на великомасштабних підприємствах. Для розгортання Zabbix в своїй локальній мережі потрібно або запустити програмних агентів (демонів), або використовувати SNMP-протокол (або інший протокол для захищеного віддаленого доступу); а для управління доведеться освоїти веб-інтерфейс на PHP. Крім того, це ПЗ надає повноцінний набір інструментів для відстеження стану апаратної частини мережі. Потребує базові знання мов Perl або Python (або будь-яких інших мов, які можна спільно використовувати з Zabbix).

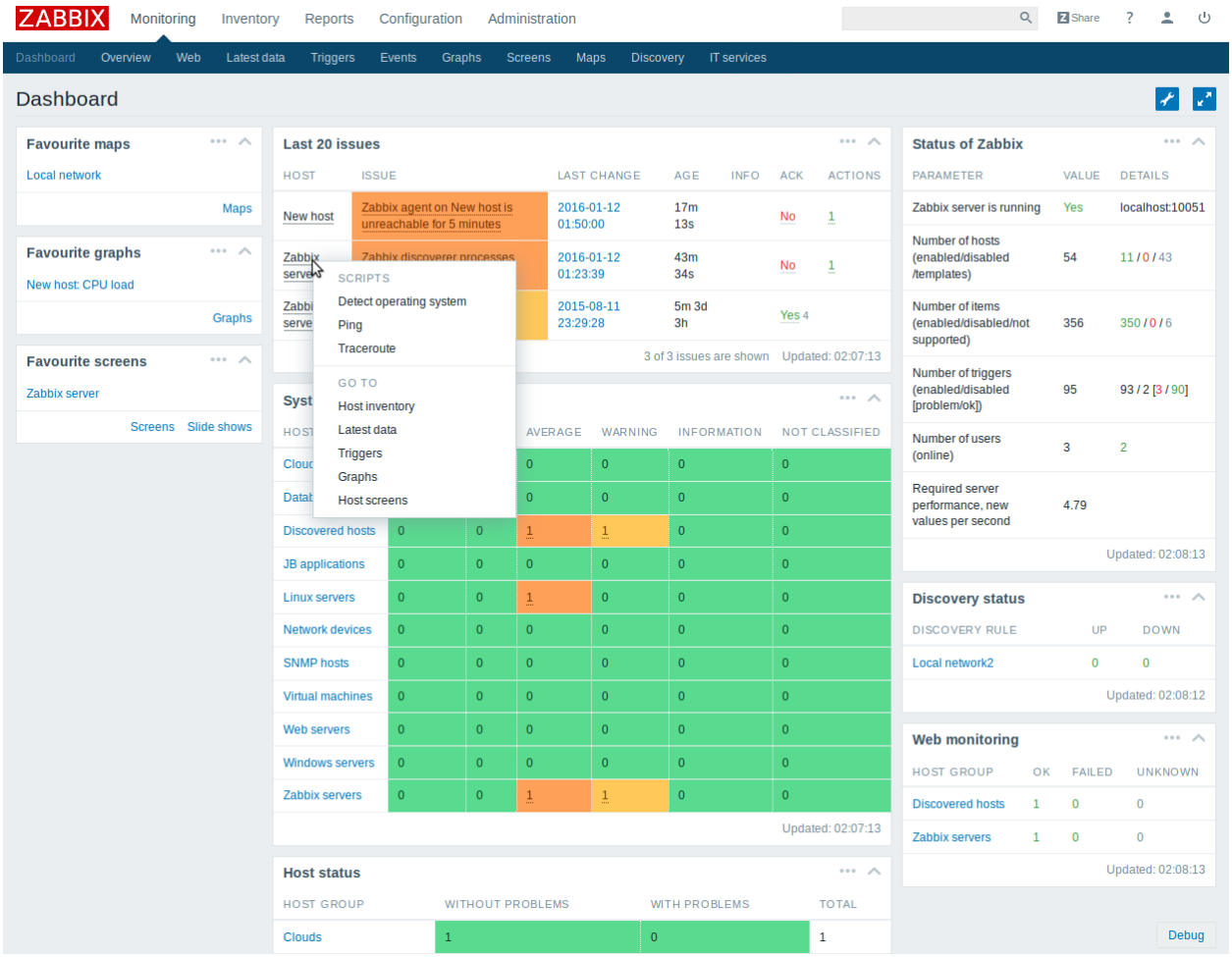

Рисунок 1.8 – Головне вікно програми Zabbix

Наведений вище аналіз показує, що більшість програм, які сьогодні пропонуються, є платними. Мають складний інтерфейс і високий поріг входу, адже вимагає від мережевих адміністраторів спеціального навчання для того, щоб почати з ним працювати. Крім того, деяким безкоштовним програмам властиві недоліки open-source рішень, такі як відсутність оновлень і погана сумісність (як з ОС, так і з ТХ пристроїв). Таким чином, актуальним є розробка власного програмного забезпечення для моніторингу та аналізу мережі, адаптованого під потреби конкретної організації. Завдяки подібному рішенню з'явиться можливість контролювати всі події, що відбуваються в межах поточної локальної мережі і своєчасно на них реагувати.

1.4 Функціональні вимоги до програмного забезпечення

Перш за все розглянемо проблеми і задачі, які повино вирішувати ПЗ, що розробляється у роботі.

Забезпечення розподілу прав доступу до ресурсів і сервісів мережі. Для організації внутрішньої безпеки мережи необхідно створити облікові записи для входу в систему і паролі для кожного співробітника та чітко розмежувати їх права. Наприклад, співробітники, технічної підтримки не зможуть отримати доступ до БД бухгалтера, відповідно бухгалтер, не зможе скористатися інформацією, з якою працює служба технічної підтримки. Розмежування доступу користувачів до глобальної мережі можна організувати на шлюзі. Виконання зазначенного кроку має сприятливо позначиться на роботі співробітників і захистити ЛОМ від шкідливого контенту і програм. З цією метою для вбудованого брандмауера, встановленого на головному маршрутизаторі мережі, пишеться правило доступу з внутрішньої мережі до зовнішньої на певні порти і доступ до них припиняється. Але для більш гнучкої і швидкої зміни налаштувань обмеження доступу цю функцію пропонується реалізувати у ПЗ, що розроблюється в роботі. Обмеження доступ конкретного користувача в Internet або до віддаленого мережевого ресурсу буде здійс-

нюватися за результатами перевірки його MAC адреси, тобто апаратної адреси мережевого адаптеру, яка є унікальним ідентифікатором конкретного мережевого вузла. Для отримання вхідного трафіку з метою подальшого аналізу, необхідно організувати прослуховування мережевого інтерфейсу. Існує певна множина програмних бібліотек, які дають можливість зчитувати мережевий трафік. Далі, потрібно виконати розбір вхідних пакетів, для вилучення з них IP і MAC адрес, розміру пакетів, а також іншої інформації. Вибір засобів для вирішення цієї проблеми виконаємо у наступних розділах роботи.

Організація моніторингу активних пристроїв. В програмі необхідно забезпечити одержання даних про активні мережеві інтерфейси, які встановлені на локальному ПК, так як саме з них буде проводитися захоплення і аналіз пакетів, а також їх подальша відправка. Перелік активного мережного обладнання можна отримати, аналізуючи MAC адреси пакетів, що надходять. Для цього потрібні спеціалізовані інструменти для прослуховування мережевих інтерфейсів, а також для «парсинга» (обробки) пакетів.

Відправлення отриманих пакетів. Пакети, які надходять на приймаючий інтерфейс, будемо його називати клієнтським інтерфейсом, після їх аналізу, підлягають подальшій відправки через другий інтерфейс, будемо його називати зовнішнім. Дану задачу можна вирішити написанням, так званого, програмного маршрутизатора. Але з деякими обмеженими можливостями. Для прийняття рішень про пересилання пакетів використовується інформація про топологію мережі і певні правила, які задані адміністратором. Існує ряд технологій в різних мовах програмування, на яких можна реалізувати подібний роутер.

Зберігання даних про мережеві пристрої. Для функціонування ПЗ і виконання ним належних функцій, необхідно зберігати інформацію про мережеві пристрої. Наприклад, при закритті програми, необхідно зберігати список дозволених MAC адрес, прикріплених до них IP адрес, а також їх обсяг трафіку. Для вирішення цього завдання можна використовувати повноцінний сервер БД, або спеціально призначені для користувача файли, з

певною структурою, які розташовуються в тій же директорії, що і виконавчий модуль програми. В цьому випадку немає потреби в установці сервера БД. Робота з такими файлами здійснюється з програми безпосередньо. У зв'язку з тим, що необхідно буде зберігати тільки кілька властивостей мережевих пристроїв, потреба у використанні сервера БД, відпадає. Але при розширенні функцій ПЗ перехід від файлової структури зберігання даних до повноцінної БД не складе труднощів.

З урахуванням вище перелічених завдань, які повинні бути реалізовані, наведемо список загальних вимог до функціоналу майбутнього програмного продукту:

- визначення і відображення на екрані адрес всіх мережевих пристроїв, які передають в мережі;
- облік трафіку по кожному мережевому пристрою (по кожній MAC адресі);
- можливість вибору із загального списку мережевих пристроїв, встановлених на ПК;
- можливість ручного введення фізичної адреси «зовнішнього» мережевого пристрою;
- можливість додавання і видалення дозволених мережевих адрес (пристроїв);
- можливість запускати і зупиняти роботу програми;
- в режимі реального часу виводити результати роботи (сканування і обробки трафіку);
- робота в операційних системах сімейства Windows;
- дружній, інтуїтивний зрозумилий для користувача інтерфейс;
- споживання мінімум системних ресурсів.

ПЗ працює на локальному комп’ютері. Ттрафік від користувачів, надходить на його другий інтерфейс, перенаправляючи трафік на перший інтер-

фейс, за яким може слідувати роутер, підключений до мережі Інтернет. Загальна схема структури мережі для роботи даного ПЗ наведена на рис. 1.9.

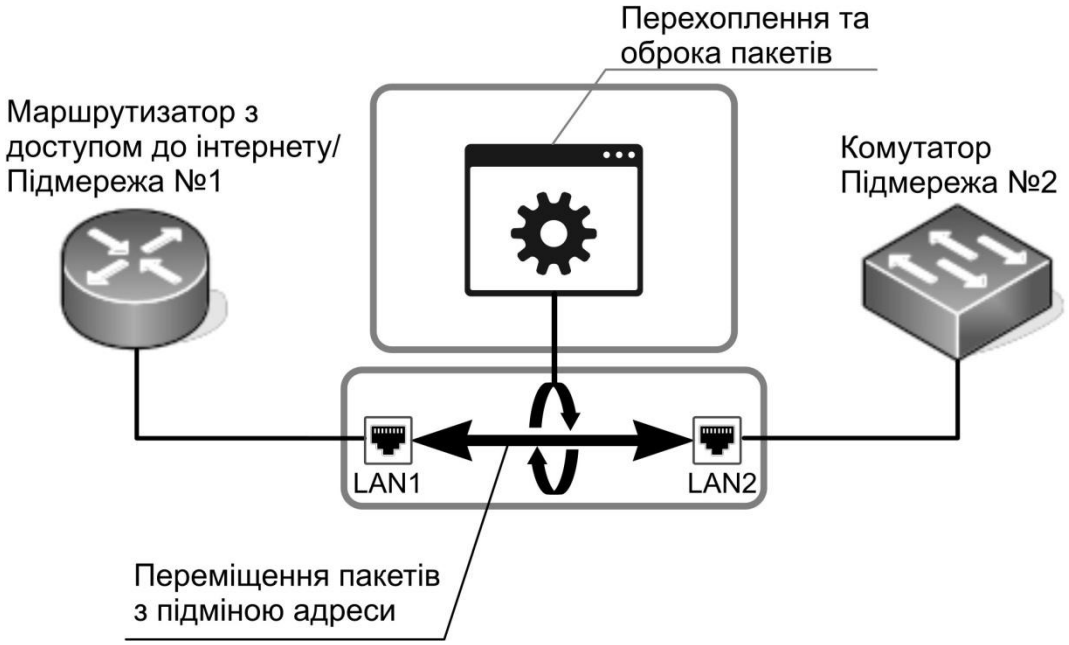

Рисунок 1.9 – Загальна схема структури мережі для функціонування ПЗ

2 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

2.1 Вибір мови програмування

Одним з важливих завдань при розробці програмного забезпечення є вдалий вибір мови програмування. Розглянемо переваги і недоліки популяриних мов програмування з метою вибору оптимального варіанту для реалізації програми. В якості кандидатів для виконання даного завдання були вибрані наступні перелічені нижче мови програмування.

Microsoft Visual C ++. Мова програмування Visual C ++ з середовища розробки Visual Studio компанії Microsoft найбільш потужний засіб розробки системних програм. Однак, розробка програм на цій мові – трудомісткий і складний процес.

Microsoft Visual Basic. Об'єктно-орієнтована мова програмування, як правило, вбудована в додатки Microsoft Office. До переваг можна віднести простоту створення нескладних додатків, а також можливість редагування і створення компонент Microsoft Office.

Delphi. Заснована на Паскалі, об'єктно-орієнтоване середовище розробки компанії Borland. Має зручні засоби для розробки віконних додатків. Крім цього є стандартні компоненти для роботи і з базами даних і з веб-сервісами.

C # .NET. C # об'єктно-орієнтована мова, як і вся платформа .NET, яка орієнтована на написання компонент. Перейнявши багато від своїх попередників – мов C ++, Java, Delphi, Модула і Smalltalk – C #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C # не підтримує множинне успадкування класів. В основі мови – легкість використання, переважна надпотужністю і швидкістю виконання. Збирач сміття, керуючи об'єктними посиланнями, автоматично звільняє пам'ять. Дає безпеку роботи з типами – один з найважливіших факторів уникнення помилок. C # дає можливість реалізації графічних додатків. Windows. Forms – це набір різних керованих бібліотек, за допомогою яких можна виконати всі необхідні для віконного програми дії, починаючи від обміну повідомленнями з операційною системою для відстеження будь-яких подій клієнтського вікна, закінчуючи діалоговими системами, зв'язком з іншими комп'ютерами по мережі і багатьма іншими можливостями. Windows Forms є ще однією з двох технологій, яка використовується в Visual C # для створення інтелектуальних клієнтських програми на основі Windows, які виконуються в середовищі .NET Framework.

Java – об'єктно-орієнтована і крос-платформна мова програмування, яка дозволяє скоротити загальний час розробки і писати повторно використовуваний код. Платформонезалежність байт-коду забезпечується наявністю віртуальних java-машин для всіх основних платформ. У комплект поставки Java входять стандартні класи, які мають достатньою функціональністю для

швидкої розробки додатків. Розвинені засоби безпеки дозволяють використовувати Java для розробки додатків, що працюють в Інтернеті. Недоліком Java є повільна швидкість роботи, обумовлена використанням JIT-компіляторів.

У табл. 2.1 представлена бальна порывняльна характеристика мов програмування, що розглядаються в роботі.

Таблиця 2.1 – Зведена порівняльна таблиця мов програмування

Параметр	C	C++	C#	Java	Python	VB.net	Assembler
Час обробки	8	8	7	6	2	6	10
Читабельність	6	6	7	7	5	10	0
Простота	2	3	6	6	10	8	0
GUI	3	4	8	7	8	10	0
Графіка (2D)	5	6	7	7	10	5	0
Графіка (3D)	8	8	7	6	1	2	0
Кросплатформність	7	7	8	10	10	2	0
Спец. процесор	10	7	0	0	0	0	5

Після проведеного аналізу, було вирішено для реалізації нашого програмного продукту використати мову програмування C #. Це обумовлено тим, що C # має широкий інструментарій для роботи з мережею, мережевими пристроями, роботою з мережевим трафіком. До неї написано безліч бібліотек, які дають можливість працювати з мережевими пристроями та трафіком, який через них проходить. Крім цього, мова має багатий інструментарій для побудови екранних форм [13][1)].

В якості середовища розробки було обране середовище Microsoft Visual Studio. Середовище розробки Visual Studio продовжує традиції корпорації Microsoft в області надання ефективних інструментальних засобів для розро-

[1)] [13] Голуб Б.М. C#. Концепція та синтаксис. Львів: Видавничий центр ЛНУ імені Івана Франка, 2006. 136 с.

бників складного ПЗ. Забезпечуючи середу розробки для всіх мов програмування, доповнену набором вікон з інтуїтивно зрозумілими інструментальними засобами, контекстної довідки, автоматизованими механізмами виконання різноманітних завдань розробки.

Visual Studio дозволяє в стислі терміни проводити професійну розробку програм різного призначення, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби для всіх платформ, що підтримуються Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe).

2.2 Бібліотеки для роботи з трафіком і мережевими пристроями

Бібліотека Pcap (Packet Capture) дозволяє створювати програми аналізу мережевих даних, що надходять на мережеву карту комп'ютера [14][1)]. Прикладом програмного забезпечення, що використовує бібліотеку Pcap, служить програма Wireshark.

Різноманітні програми моніторингу та тестування мережі, сніфери використовують цю бібліотеку. Вона призначена для використання спільно з мовами C / C ++ / C #, а для роботи з бібліотекою на інших мовах, таких як Java, .NET, використовують обгортки. Для Unix-подібних систем це бібліотека libpcap, а для Microsoft Windows – WinPcap. Програмне забезпе-

[1)] [14] Бібліотека Pcap. Вікіпедія. URL: https://ru.wikipedia.org/wiki/Pcap (дата звернення 13.05.2019)

чення мережевого моніторингу може використовувати libpcap або WinPcap, щоб захопити пакети, які переміщуються мережєю, і (в новіших версіях) для передачі пакетів в мережі. Libpcap і WinPcap також підтримують збереження захоплених пакетів в файл і читання файлів, що містять збережені пакети.

Програми, написані на основі libpcap або WinPcap, можуть захопити мережевий трафік, аналізувати його. Файл захопленого трафіку зберігається в форматі, зрозумілому для додатків, що використовують Pcap. Іншими словами, для того щоб перехоплювати пакети, мережева карта повинна працювати в так званому режимі promiscuous-mode.

Для роботи в такому режимі потрібна підтримка на рівні драйверів, яку і забезпечує спеціальна бібліотека WinPcap. Склад WinPcap:

- фільтр пакетів на рівні ядра;
- низкоуровневая DLL (packet.dll);
- високорівнева системно-незалежна бібліотека (wpcap.dll).

Єдиний недолік даної бібліотеки в тому, що вона працює не з усіма нестандартними адаптерами (Wi-Fi-картками, VPN і т.д).

Бібліотеки SharpPcap і Packet. Net SharpPcap – бібліотека для .NET, яка дозволяє перехоплювати пакети. По суті, це обгортка над бібліотекою Pcap, яка використовується в багатьох популярних продуктах. Наприклад, сниффер Wireshark, IDS Snort.

З SharpPcap також поставляється чудова бібліотека для парсинга пакетів – Packet.Net. Packet.Net підтримує наступні протоколи: Ethernet, LinuxSLL, Ip (IPv4 and IPv6), Tcp, Udp, ARP, ICMPv4 і ICMPv6, IGMPv2, PPPoE, PTP, Link Layer Discovery Protocol (LLDP), Wake-On-LAN (WOL) .

2.3 Віртуальна машина VMware

VMware Workstation – програмне забезпечення віртуалізації, призначене для комп'ютерів x86-64 операційних систем Microsoft Windows і Linux

[15][1)]. Дозволяє користувачу встановити одну або більше віртуальних машин на один фізичний комп'ютер і запускати їх паралельно з ним. Кожна віртуальна машина може виконувати свою операційну систему, включаючи Microsoft Windows, Linux, BSD, і MS-DOS. VMware Workstation розроблена і продається компанією VMware, підрозділом EMC Corporation. VMware Workstation підтримує мости з мережним адаптером реального комп'ютера, а також створення спільних папок з віртуальною машиною. Програма може монтувати реальні CD або DVD диски або ISO образи в віртуальні оптичні приводи, при цьому віртуальна машина буде вважати, що приводи справжні. Віртуальні жорсткі диски зберігаються в файлах .vmdk. VMware Workstation в будь-який момент може зберегти поточний стан віртуальної машини (знімок). Дані знімки пізніше можуть бути відновлені, що повертає віртуальну машину в збережений стан. VMware Workstation включає в себе можливість об'єднувати кілька віртуальних машин в групу, яку можна включати, вимикати, призупиняти або відновлювати як єдиний об'єкт, що є корисним для тестування технологій клієнт-сервер. Віртуальні машини на платформі VMware дозволяють користувачам створювати різні комбінації віртуальних систем, що працюють за різними принципами мережевого взаємодії. Основою мережі VMware є такі компоненти:

– віртуальні комутатори (Virtual Switches);

– віртуальні мережеві інтерфейси (Virtual Ethernet Adapters);

– віртуальний міст (Virtual Bridge);

– вбудований DHCP-сервер;

– пристрій трансляції мережевих адрес (NAT, Network Address Translation).

Фундаментальним елементом мережевої взаємодії в VMware Workstation є віртуальний комутатор. Він забезпечує мережеву взаємодію віртуальних машин на манер фізичного пристрою: на віртуальному

[1)] [15] VMware Workstation. Вікіпедія. URL: https://ru.wikipedia.org/wiki/VMware_Workstation (дата звернення 13.05.2019)

комутаторі є порти, до яких можуть бути прив'язані віртуальні мережеві інтерфейси віртуальних машин, а також інші компоненти віртуальної інфраструктури в межах хоста. Кілька віртуальних машин, підключених до одного віртуального комутатора, належать одній підмережі. Віртуальний міст являє собою механізм, за допомогою якого відбувається прив'язка фізичного мережевого адаптера комп'ютера до віртуальних мережних інтерфейсів. Вбудований DHCP-сервер VMware дозволяє віртуальним машинам автоматично отримувати IP-адресу в своїй підмережі, а віртуальний NAT-пристрій забезпечує трансляцію мережевих адрес при спілкуванні віртуальних машин із зовнішньою мережею. Продукти VMware Workstation надають користувачам можливість призначити віртуальній машині один з трьох базових типів мережевої взаємодії для кожного з віртуальних мережевих адаптерів:

- Bridged;
- Host-only;
- NAT.

Bridged Networking – тип мережевої взаємодії, який дозволяє прив'язати мережевий адаптер віртуальної машини до фізичного мережевого інтерфейсу комп'ютера, що дає можливість розділяти ресурси мережевої карти між хостовою і віртуальною системою. Віртуальна машина з таким типом мережевої взаємодії буде вести себе по відношенню до зовнішньої мережі хостовой системи як незалежний комп'ютер. Структура Bridged Networking приведена на рис 2.1.

Networking Host-Only Networking – тип мережевої взаємодії, оптимальний для цілей тестування програмного забезпечення, коли потрібно організувати віртуальну мережу в межах хоста, а віртуальним машинам не потрібно вихід в зовнішню мережу. Структура Host-Only Networking приведена на рис 2.2.

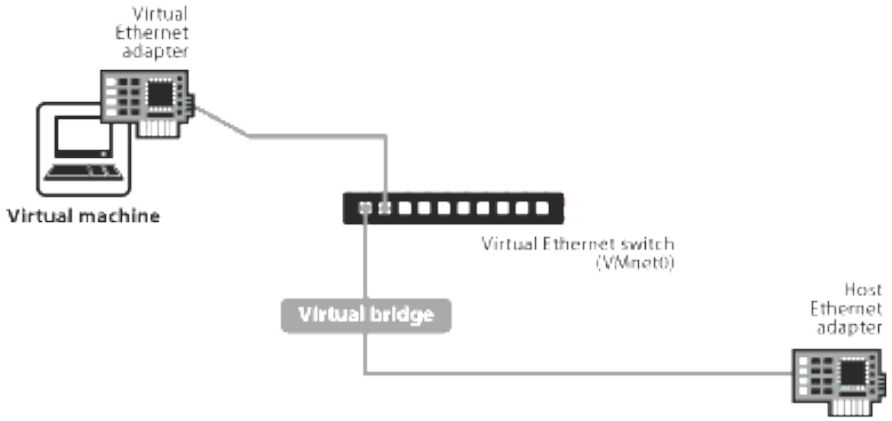

Рисунок 2.1 – Структура мережевої взаємодії в режимі Bridged

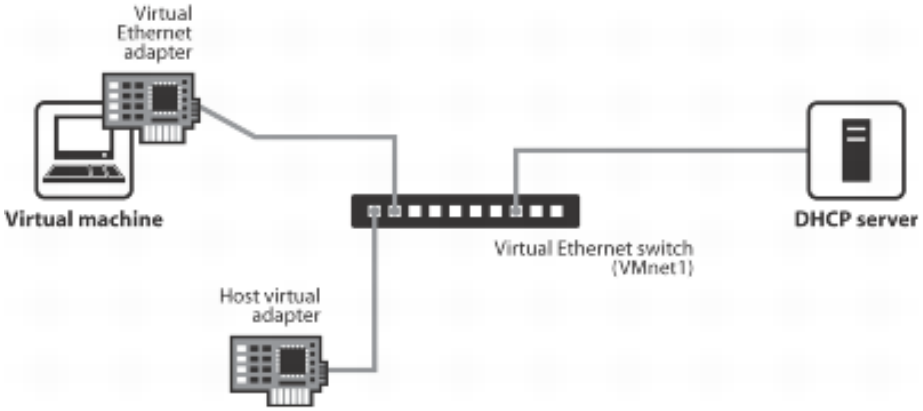

Рисунок 2.2 – Структура Host-Only Networking

NAT Networking. Цей тип мережевої взаємодії дуже схожий на Host-Only, за одним винятком: до віртуального комутатора VMnet8 підключається пристрій трансляції IP-адрес (NAT). До цього комутатора також підключається DHCP-сервер, що роздає віртуальним машинам адреси з заданого діапазону (за замовчуванням 192.168.89.128 – 192.168.89.254) і, безпосередньо, самі віртуальні машини. NAT-пристрій дозволяє здійснювати трансляцію IP-адрес, що дозволяє віртуальним машинам ініціювати з'єднання в зовнішню мережу, які дають при цьому механізму доступу до віртуальних машин ззовні. Структура NAT Networking приведена на рис 2.3.

При створенні і налаштування стенду, будемо використовувати підключення Host-Only.

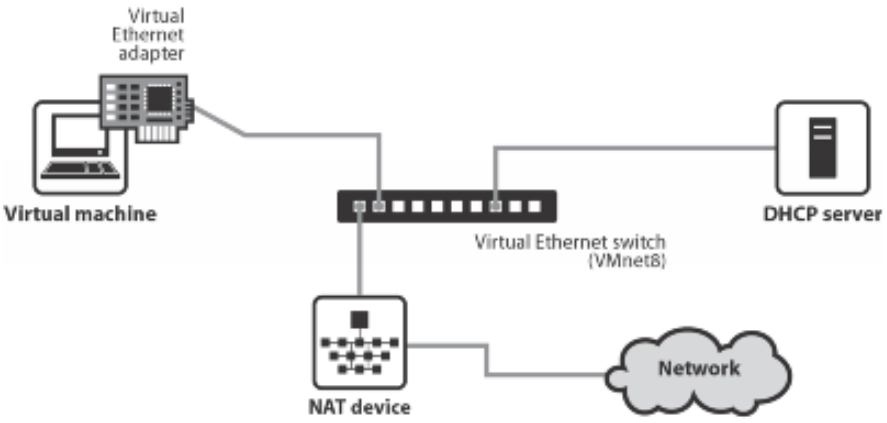

Рисунок 2.3 – Структура NAT Networking

2.4 Налаштування стенду

Для налагодження і тестування розроблюваного програмного продукту, необхідно було створити і налаштувати стенд, який представляє собою локальну мережу, що складається з трьох ПК, з встановленою програмою VMware workstation. На клієнтських і центральних машинах встановлена ОС Windows 7. Крім цього, на центральному хості були встановлені: Visual Studio 2017; аналізатор пакетів Wireshark, з бібліотекою WinPcap; бібліотеки SharpPcap.dll і Pcap.NET.dll.

Кожна з клієнтських машин має свою MAC адресу та призначені IP адресу і адресу шлюза. Налаштування хостів мережі представлені на рис. 2.4. Основний хост з двома інтерфейсами так само отримав мережеві адреси для двох мережевих адаптерів.

Підключивши перший хост до першого мережного інтерфейсу головного хосту, а другий – до другого інтерфейсу, отримали стенд з мінімальною кількістю засобів, необхідних для розробки і тестування програмного забез-

печення. Після підключення, зв'язок між хостами в мережі була перевірений за допомогою команди Ping.

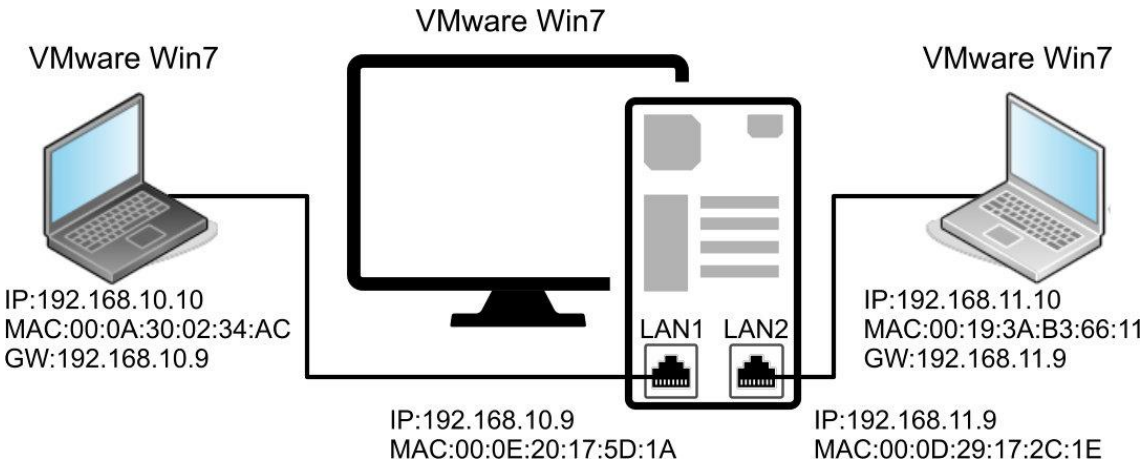

Рисунок 2.4 – Схема налаштування стенду для розробки ПЗ

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

3.1 Етапи розробки програмного забезпечення

Розробка будь-якого ПЗ складається з декількох етапів, грамотна реалізація яких є обов'язковою умовою для отримання хорошого результату. Стисло розглянемо кожну стадію методології розробки ПЗ [16][1)].

Аналіз вимог. Першим етапом розробки ПЗ є процедура проведення всебічного аналізу вимог до створюваного ПЗ, яка дозволяє визначити ключові цілі та завдання кінцевого продукту. В рамках цієї стадії відбувається обговорення деталей проекту, яке допомагає більш чітко сформулювати вимоги до ПЗ. Результатом проведеного аналізу стає формування основного регламенту, на який спиратиметься виконавець у своїй роботі – технічного завдання (ТЗ) на розробку ПЗ. ТЗ повинно повністю описувати

[1)] [16] Кармайкл Э., Хэйвуд Д. Быстрая и качественная разработка программного обеспечения. М.: Издательский дом «Вильямс», 2003. 403 с.

завдання, які поставлені перед розробником, і характеризувати кінцеву мету проекту.

Проектування. Наступний етап в розробці ПЗ – стадія проектування, тобто моделювання теоретичної основи майбутнього продукту. Найсучасніші засоби програмування дозволяють частково об'єднати етапи проектування та кодування, тобто технічної реалізації проекту, будучи заснованими на об'єктно-орієнтованому підході, але повноцінне планування вимагає більш ретельного і скрупульозного моделювання. Якісний аналіз перспектив і можливостей створюваного продукту стане основою для його повноцінного функціонування і виконання всього комплексу покладених на ПЗ завдань. Однією із складових частин етапу проектування, наприклад, є вибір інструментальних засобів і операційної системи, яких сьогодні на ринку присутня дуже велика кількість. В рамках даного етапу треба здійснити:

– оцінку результатів проведеного спочатку аналізу і виявлених обмежень;

– пошук критичних ділянок проекту;

– формування остаточної архітектури створюваної системи;

– аналіз необхідності використання програмних модулів або готових рішень сторонніх розробників;

– проектування основних елементів продукту – моделі бази даних, процесів і коду;

– вибір середовища програмування і інструментів розробки, затвердження інтерфейсу програми, включаючи елементи графічного відображення даних;

– визначення основних вимог до безпеки ПЗ, що розробляється.

Кодування. Наступним кроком є безпосередня робота з кодом, спираючись на обрану в процесі підготовки мову програмування. Описувати особливості і тонкощі самого трудомісткого і складного етапу навряд чи варто, досить вказати, що успіх реалізації будь-якого проекту безпосередньо за-

лежить від якості попереднього аналізу і оцінки конкуруючих рішень, з якими створюваній програмі треба бути «боротися» за право називатися кращою в своїй ніші. Кодування може відбуватися паралельно з наступним етапом розробки – тестуванням програмного забезпечення, що допомагає вносити зміни безпосередньо по ходу написання коду. Рівень і ефективність взаємодії всіх елементів, задіяних для виконання сформульованих завдань розробником, на поточному етапі є найважливішим – від нього залежить якість реалізації проекту.

Тестування та налагодження. Дозволяє ліквідувати огріхи програмування і домогтися кінцевої мети – повнофункціональної роботи розробленої програми. Процес тестування дозволяє змоделювати ситуації, при яких програмний продукт (ПП) перестає функціонувати. Етап алагодження локалізує і виправляє виявлені помилки коду, «вилизуючи» його до практично ідеального стану. Ці два етапи займають не менше 30% часу, що витрачається на весь проект.

Впровадження. Процедура впровадження ПЗ в експлуатацію є завершальною стадією розробки і нерідко відбувається спільно з налагодженням системи. Як правило, введення в експлуатацію ПЗ здійснюється в три етапи:

– первісне завантаження даних;
– поступове накопичення інформації;
– виведення створеного ПЗ на проектну потужність.

Ключовою метою поетапного впровадження розробленої програми стає поступове виявлення не виявлених раніше помилок і недоліків коду. В рамках цього етапу розробки ПЗ можна зіткнутися з низкою досить вузького спектра помилок, пов'язаних з частковою неузгодженістю даних при їх завантаженні в БД, а також зривів виконання програмних процедур в зв'язку з застосуванням доступу на багато користувачів. Саме на цій стадії викристалізовується остаточна картина взаємодії користувача з програмою, а також визначається ступінь лояльності останнього до розробленого інтерфейсу.

Створення навіть невеликого і технічно простого ПЗ залежить від чіткого виконання кожної фази розробки. Чіткий план виконання необхідних заходів з зазначенням кінцевої мети стає невід'ємною частиною роботи розробників. Тільки правильно складене ТЗ дозволить домогтися потрібного результату і здійснити розробку по-справжньому якісного і конкурентного ПЗ для будь-якої платформи – серверної, стаціонарної або мобільної. Невід'ємною частиною завершального етапу розробки ПЗ також є подальша технічна підтримка створеного ПП в процесі його експлуатації.

3.2 Розробка графічного інтерфейсу користувача

Для створення графічного інтерфейса користувача у середовищі Visual Studio 2017 був створений проект WinForms. Консольний варіант програми не підходить для виконання поставленого завдання, через відсутність інструментів для управління ПЗ. Структурно, програма буде складатися з трьох головних складових:

- Інтерфейс програми буде реалізований у вигляді графічного вікна, на якому розміщені елементи управління програми, а також поля для введення і виведення інформації.
- Логічна частина програми – це програмний код, який відповідає за логіку роботи програми. Іншими словами – це всі складові функції програми, які відповідають за її функціонал і роботу.
- Допоміжні бібліотеки, необхідні для доступу до мережних інтерфейсів і для перехоплення трафіку (пакетів). В проекті використані допоміжні бібліотеки (Pcap.dll, SharpPcap.dll, Pcap.NET.dll). Для цього у вкладці "References" додано посилання на вищевказані бібліотеки. Для того щоб додати посилання, необхідно натиснути правою кнопкою миші на пункт "References", і в меню на пункт "Add Reference ...", як показано на рис. 3.1.

Після чого, на екрані з'явиться менеджер пошуку і вибору посилань як показано на рис. 3.2.

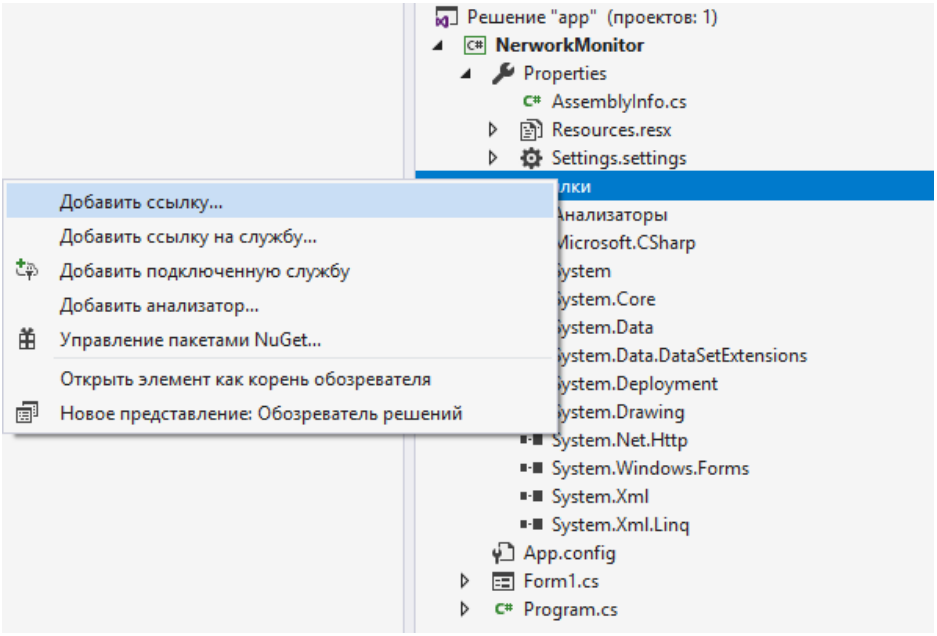

Рисунок 3.1 – Крок додавання посилань на допоміжні бібліотеки

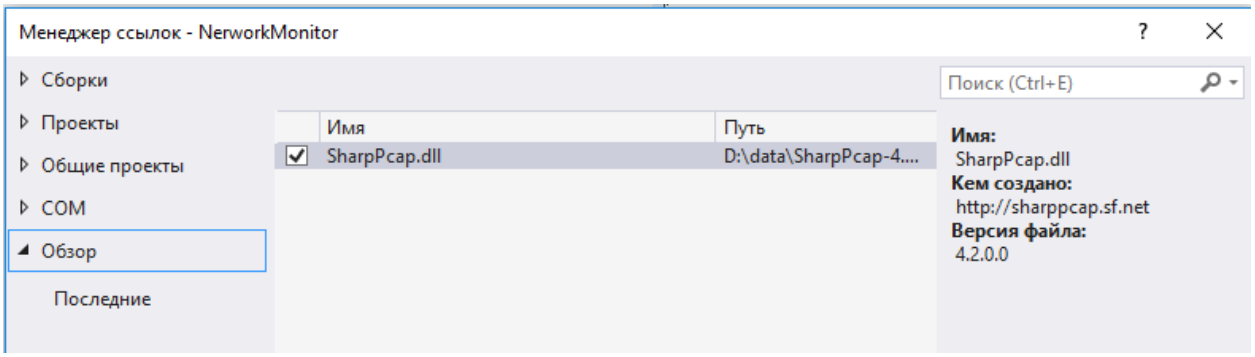

Рисунок 3.2 – Вибір і додавання посилань

На форму були додані наступні компоненти:

– cbFirstDev – компонент comboBox, для вибору мережевого підключення з боку клієнтів;

– cbLastDev – компонент comboBox, для вибору мережевого підключення з боку другої підмережі (роутера з Інтернет);

– tbDest – компонент textBox, для введення MAC адреси роутера підключеного до Інтернет або інший підмережі;

– dgOther – компонент dataGridView, для виведення списку всіх мережевих пристроїв, від яких йде мовлення в мережі (через другий інтерфейс основної машини);

– dgClients – компонент dataGridView, для виведення списку всіх дозволених мережевих пристроїв (клієнтів). Терміну складається з: MAC адреси, IP адреси, обсягу трафіку (в байтах);

– btRefresh – компонент button, для оновлення списків dgOther і dgClients;

– AddClient – компонент button, для додавання обраного клієнта зі списку dgOther;

– button2 – компонент button, для видалення вибраного клієнта зі списку dgClients;

– btstart – компонент button, для запуску процесу сканування мережі, перехоплення пакетів та інших процесів;

– btstop – компонент button, для зупинки всіх процесів.

Так само, для наочності та опису всіх компонент були додані компоненти label. Для періодичного оновлення списків dgOther і dgClients в проект доданий компонент timer. Після додавання всіх необхідних компонентів і їх позначення за допомогою компонентів label, форма набуває вигляду як показано на рис. 3.3.

3.3 Розробка UML діаграми класів

Універсальна мова моделювання (Unified Modelling Language або UML) – це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей орієнтованих на об'єкти систем програмного забезпечення. Це спосіб візуалізації програмного забезпечення з викорис-

танням набору діаграм. UML не є методом розробки, тобто, в конструкціях цієї мови не повідомляється про те, в якій послідовності повинні виконуватися етапи моделювання і не надається інструкцій з побудови системи, але ця мова допомагає наочно переглядати компонування системи та полегшує співпрацю з іншими її розробниками. Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення [17][1)].

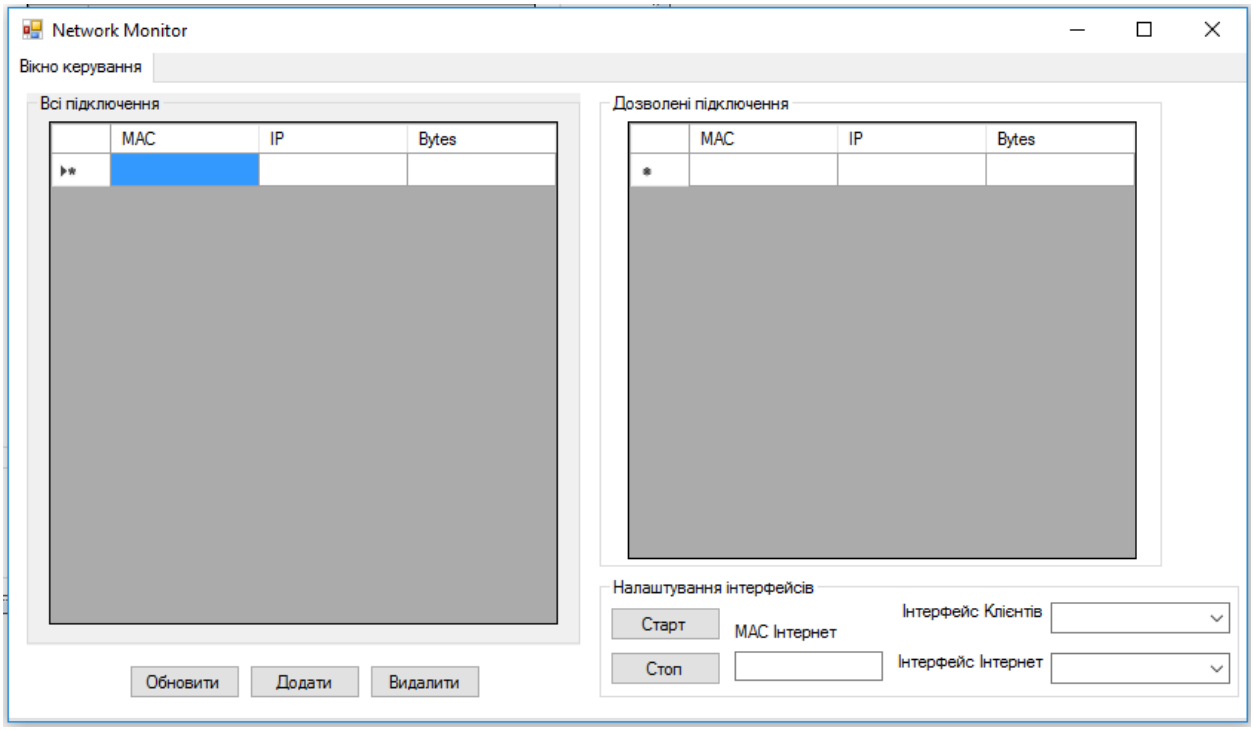

Рисунок 3.3 – Інтерфейс програми

UML розроблена для розробки структури орієнтованого на об'єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML

[1)] [17] Дудзяний І.М. Об’єктно-орієнтоване моделювання програмних систем. Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. 108 с.

використовуються для побудови діаграм, які відповідають певної частини системи або точці зору на систему.

Діаграмою класів в термінології UML називається діаграма, на якій зображений набір класів, які не мають явного відношення до проектування бази даних, а також зв'язків між цими класами. Крім того, діаграма класів може включати коментарі обмеження. Обмеження можуть неформально задаватися природною мовою або ж можуть формулюватися мовою об'єктних обмежень OCL (Object Constraints Language). Інакше кажучи, діаграма класів може відображати тільки імена класів або імена і відповідні атрибути класів, або імена, атрибути і операції (методи) класів.

Клас – це опис набору об'єктів з однаковими атрибутами, операціями, зв'язками і семантикою. Кожен клас повинен володіти ім'ям, який вирізняє його від інших класів. Ім'я – це текстовий рядок. Ім'я класу може складатися з будь-якого числа букв, цифр і розділових знаків (за винятком двокрапки і крапки) і може записуватися в кілька рядків.

Атрибут (властивість) – це іменована властивість класу, яка описує діапазон значень, які може приймати примірник атрибута. Клас може мати будь-яке число атрибутів або не мати жодного. В останньому випадку блок атрибутів залишають порожнім. Атрибут представляє деяку властивість сутності, що моделюється, якою володіють всі об'єкти даного класу. Ім'я атрибута, як і ім'я класу, може являти собою текст. На практиці для іменування атрибута використовуються одне або кілька коротких іменників, що виражають якесь властивість класу, до якого належить атрибут.

Атрибути можуть бути такими, типи значень яких вважаються заздалегідь визначеними в UML, як: розмір, площа, кут, видимість. Останній атрибут може мати наступні значення:

− загальний (public) означає, що операція класу доступна для кожного об’єкту системи з будь-якого місця програми;

– захищений (protected) означає, що операція класу може бути доступна тільки для тих об’єктів системи, які є екземплярами цього класу або екземплярами його спадкоємців;

– приватний (private) означає, що операція класу може бути доступна тільки для тих об’єктів системи, які є екземплярами цього класу – тобто класу, в якому вона визначена.

Спроектована діаграма класів для програмної системи наведена на рис. 3.4.

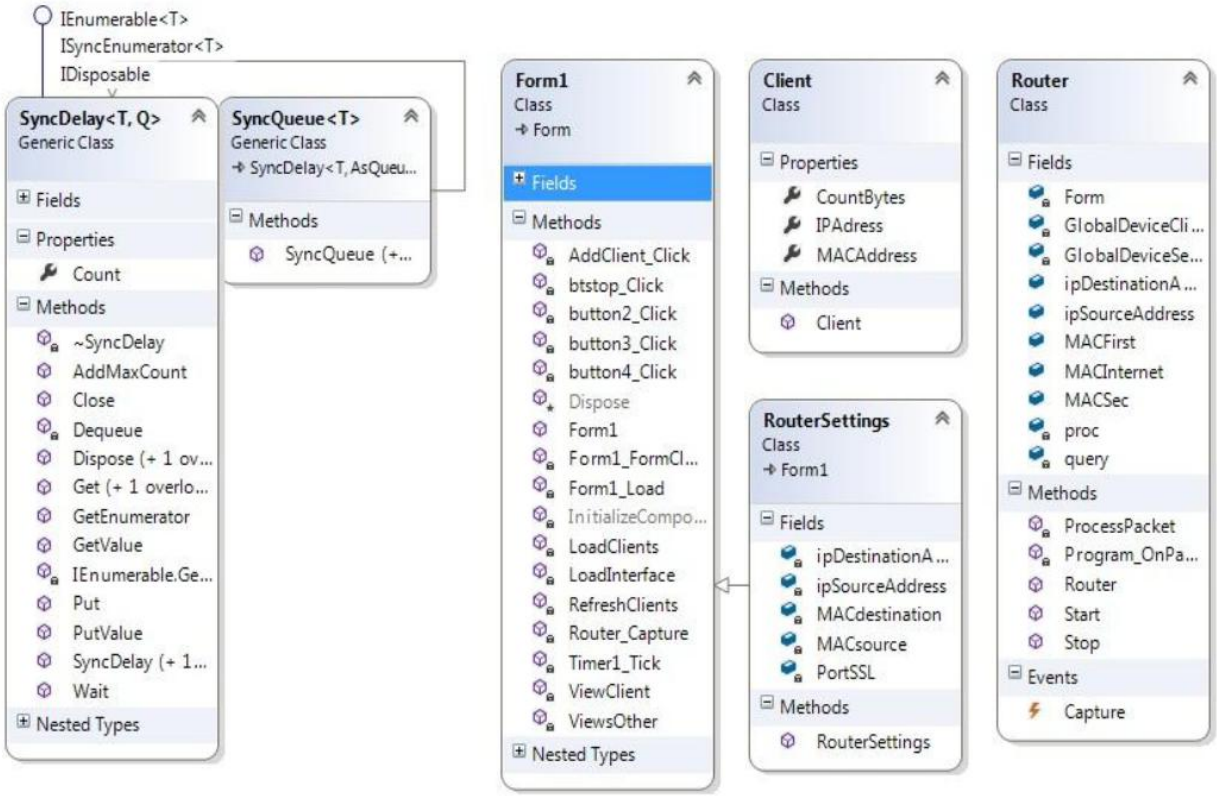

Рисунок 3.4 – Діаграма класів

В ході роботи виділено такі класи:

– Form1 – клас екранної форми;

– Client – клас відомостей про клієнтів;

– Router – клас відповідає за захоплення і обробку пакетів;

– SyncDelay, SyncQueue – класи роботи з чергами.

3.4 Реалізація логіки роботи програми

Перш за все в програмі були підключені бібліотеки SharpPcap.dll і Pcap.NET.dll, які надають доступ до поточних мережевих підключень, пристроїв, а також дають можливість перехоплювати мережевий трафік і обробляти і змінювати пакети. Також, була підключена бібліотека System.Net.NetworkInformation. Підключення вищезазначених бібліотек командою using показано на рис. 3.4.

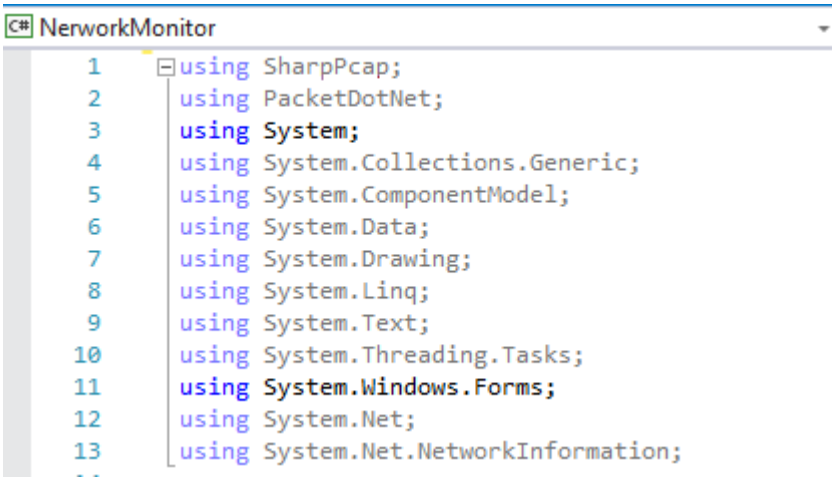

Рисунок 3.4 – Підключення бібліотек в проекті

Для зберігання даних про дозволених мережеві інтерфейси (клієнтів) для перемещення пакетів, використовується спеціальний текстовий файл "Clients.txt". Формат записів файлу: [MAC адреса]: [IP адреса]: [обсяг трафіку].

Основні розроблені функції програми наведені у додатку А. Розглянемо більш детально призначення окремих функцій.

Функція Form1_Load() є обробником події завантаження форми, тобто функцією, яка відпрацьовується при першому завантаженні форми. В тілі

функції передбачена перевірка на наявність файлу "Clients.txt". Якщо такий файл відсутній, то він створюється в директорії, де знаходиться бінарний (exe) файл. Якщо такий файл існує, виконується функція LoadInterface(), потім функція LoadClients() і в підсумку, функція ViewClient().

За допомогою методу CaptureDeviceList.Instance() можна отримати список мережевих підключень (пристроїв) і записати списки пристроїв в компоненти cbFirstDev і cbLastDev. Далі виконується функція LoadClients().

Перш за все, з файлу "Clients.txt" зчитуються дані про клієнтів і записуються в об'єкт класу Client.

Після того як всі записи з файлу "Clients.txt" були прочитані і записані в список Clients, викликається функція ViewClient() для виведення списку клієнтів в форму в компонент dgClients.

Далі, при натисканні на кнопку "Старт" викликається функція обробник події button3_Click().

Після запуску, функція, підключившись, до обраних мережних інтерфейсів, запускає процес перехоплення і обробки пакетів. За таймером здійснюється оновлення списку клієнтів.

Функція ViewsOther() здійснює запис в компонент dgOther списку всіх клієнтів, від яких надходить трафік за час, який працює процес перехоплення. Працює при вибраному циклічно за таймером. Таймер встановлений на 20 секунд.

Функція AddClient_Click() – обробник події натискання на кнопку "Додати", здійснює копіювання вибраного рядка з компонента dgOther в файл "Clients.txt", а звідти в компонент dgClients.

Функція button4_Click () при натисканні кнопки button4, здійснює оновлення списків dgClients, dgOther і списку Clients, шляхом запуску функції RefreshClients().Оновлює список клієнтів в файлі "Clients.txt".

При закритті форми, виконується функція this.Dispose (). Для того, щоб завершити (припинити) процес перехоплення, необхідно натиснути на кноп-

ку "Стоп". Для неї реалізована функція обробки натискання btstop_Click(). Функція зупиняє процес перехоплення.

У програмі реалізовані спеціалізовані стек і черги, для простоти реалізації процесу перехоплення і обробки пакетів. В окремому файлі "Router.cs", реалізований клас class Router. Для якого написаний користувальницький конструктор і наступні методи:

– public void Start (ICaptureDevice first, ICaptureDevice second) – приймає на вхід два мережевих пристроя, з яких буде зчитуватися і відправлятися мережевий трафік. Цей метод здійснює підміну MAC адрес і перенаправляє пакети (здійснює функцію роутера), саме завдяки йому, є можливість відправляти пакети з однієї підмережі, в іншу.

– public void Stop (Router router) – приймає на вхід об'єкт типу Router. Здійснює зупинку захоплення пакетів.

– void Program_OnPacketArrival10 (object sender, CaptureEventArgs e) – метод, що обробляє події отримання нового пакету інтерфейсом.

– private void ProcessPacket () – здійснює обробку пакетів рівня IP.

3.5 Тестування роботи програми

Перед тим як запускати програму, необхідно переконатися, що на локальній машині існують два мережевих пристроя, що вони підключені і працюють правильно. Необхідно перевірити підключення до цих пристроїв клієнтів, інормації про які буде фільтруватися і для яких буде здійснена політика розмежування доступу в зовнішню мережу. Зовнішньою мережею може бути як локальна мережа, адреса підмережі якої, відрізняється від першої, так і роутер, який має доступ до мережі інтернет, як це показано на рис. 3.5.

Наступним кроком, є перевірка зв'язку підмережі №1 з відповідним інтерфейсом центральної машини, а підмережі №2 з мережевим пристроєм, в який вона подключена на центральній машині. Це здійснюється командою

"ping". Після того як всі вузли пов'язані, можна приступати до запуску програми.

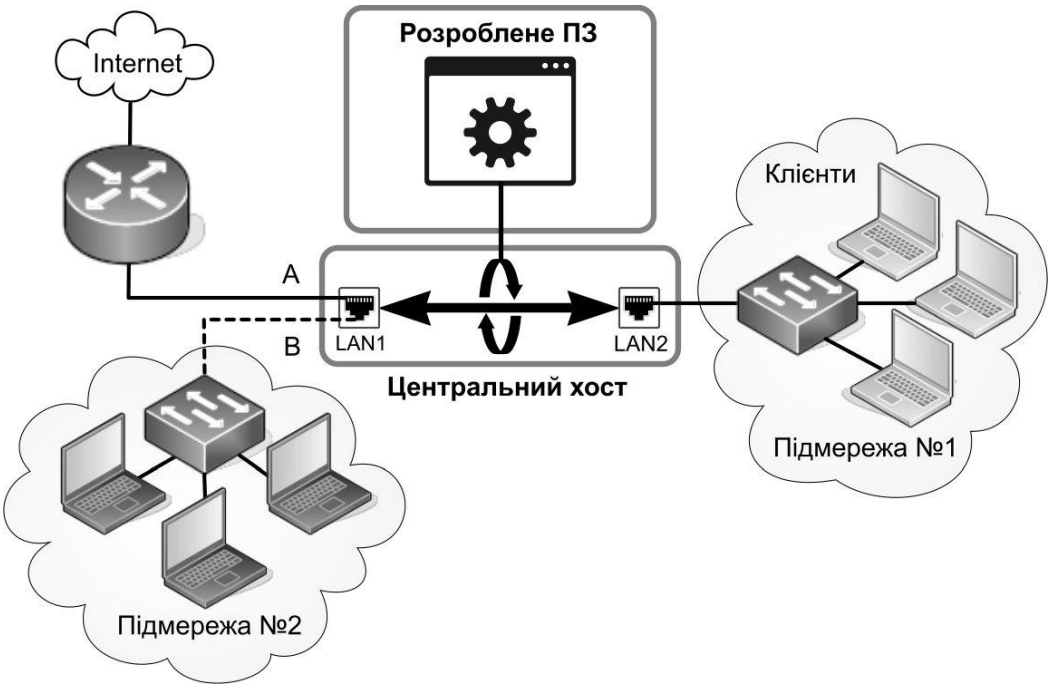

Рисунок 3.5 – Схема варіантів структури мережі

До мінімальної конфігурації центрального ПК є певні вимоги:

- процесор: Intel Core 2 Duo, AMD Sempron чи вище;
- обсяг ОП: не менше 256-512 Мб вільної пам'яті;
- обсяг жорсткого диска: не менше 128 Мб вільного місця;
- мінімум два мережевих інтерфейсу, з підтримкою режиму прослуховування мережі;
- ОС не нижче Windows 7.

Обов'язковими вимогами є:

- версія NET.FrameWork не нижче 4.5.2;
- версія WinPcap не нижче 4.1.3;
- наявність бібліотек SharpPcap.dll, Pcap.NET.dll в директорії з виконуваним модулем програми;

– мережеві екрани на на центральній машині, повинні бути відключені в обов'язковому порядку.

Переконавшись, що всі необхідні бібліотеки на місці, можна запускати програму. Після запуску, першою дією є запис MAC адреси свого пристрою на який буде здійснюватися перенаправлення трафіку (підмережа №1). Це необхідно зробити в текстовому полі «MAC Інтернет». Потім, в списках "Інтерфейс Клієнтів" і "Інтерфейс Інтернет", необхідно вибрати відповідні мережеві пристрої. Їх назви в списках будуть відповідати назвам, які вони мають в ОС. Інтерфейс програми під час її роботи представлений на рис.3.6.

Для запуску процесу моніторингу та фільтрації, необхідно натиснути на кнопку "Старт". Після запуску, очікуємо появи перших адрес мережевих пристроїв, пакети від яких проходять через мережевий інтерфейс з боку підмережі №2. Інформація про ці мережеві пристрої з'являтиметься в списку "Всі підключення". А сам запис про влаштування матиме вигляд переліку MAC адреси, IP адреси і кількості байтів.

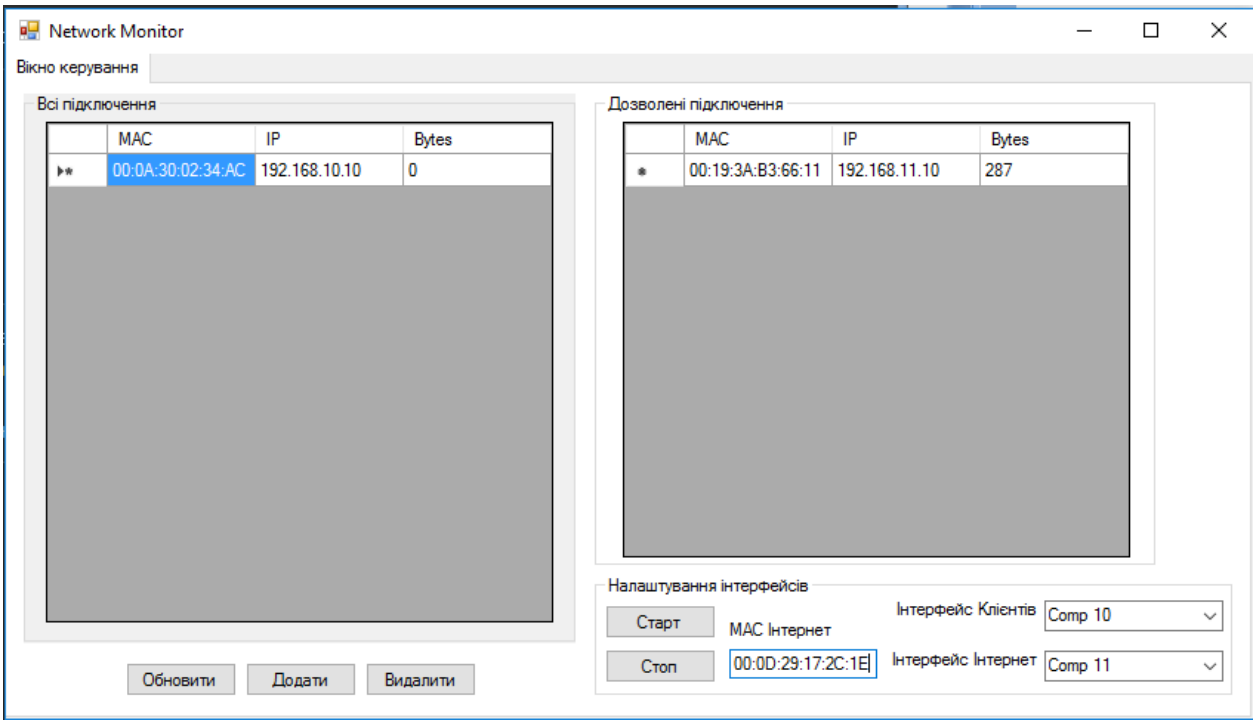

Рисунок 3.6 – Вікно програми під час роботи

Для того, щоб додати мережевий інтерфейс із загального списку в список "Дозволені підключення", необхідно його вибрати з списку "Всі підключення" і натиснути кнопку "Додати". Після чого, мережевий пристрій потрапляє в список "Дозволені підключення". Так само, після того як пристрій потрапляє в список "Дозволені підключення", для нього здійснюється підрахунок трафіку. Для того, щоб не чекати оновлення інформації в списку за таймером, можна скористатися кнопкою "Обновити", і списки миттєво оновляться.

Для видалення пристрою зі списку "Дозволені підключення", необхідно натиснути кнопку "Видалити". І обраний мережевий пристрій знову потрапить в список "Всі підключення". Для зупинки процесу моніторингу та фільтрації, необхідно натиснути на кнопку "Стоп". Для повторного запуску, досить буде знову натиснути на кнопку "Старт". Якщо закрити програму і знову її запустити, то список дозволених клієнтів завантажитися з файлу. Програма, при закритті, запам'ятовує списки дозволених клієнтів.

Тестування програмного забезпечення – це [18][1)]:

− процес дослідження ПЗ з метою отримання інформації про якість продукту;

− процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином;

− оцінка системи з тим, щоб знайти відмінності між тим, якою система повинна бути і якою вона є.

У широкому сенсі, тестування – це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування і аналіз отриманих результатів.

[1)] [18] Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб. : Питер, 2004. 320 с.

Системне тестування – це тестування програми в цілому. Для невеликих проектів це, як правило, ручне тестування.

Об'єктом випробувань є наш ПП. Перед запуском ПЗ проводилася спроба перевырити з’єднання однієї під мережі з іншою командою "ping". Для цього, з машини 192.168.11.10, відіслали команду "ping" на вузол 192.168.10.10. Але відповіді від цього вузла не було і це вірно. Так як дані хости перебувають в різних підмережах. Після чого, була запущена програма. Введено MAC адреса зовнішнього вузла і обрані два мережевих інтерфейса, клієнтський і зовнішній. Потім, знову послали команду "ping". Тепер, це потрібно було для того, щоб наш клієнт з'явився в загальному списку мережевих пристроїв. Як тільки він з'явився, вибравши його і натиснувши на кнопку "Додати", пакети "ping" стали доходити до вузла 192.168.10.10, а відповіді, надходити назад до 192.168.11.10.

Тестування всіх програмних модулів проводилося на етапі налагодження з усуненням виявлених помилок.

Для тестування користувальницького інтерфейсу і реалізованих функцій (логіки), був розроблений набір можливих штатних і позаштатних дій користувача:

- спроба підключення зовнішнього мережевого вузла з неіснуючою MAC адресою;
- спроба запуску програми з невибраними мережевими інтерфейсами на локальній машині;
- запуск програми без файлу клієнтів;
- вказівка неправильного формату даних у файлі з клієнтами;
- перевірка роботи всіх кнопок;
- перевірка коректності відображення списків;
- перевірка коректності роботи таймера;
- перевірка коректності перехоплення і проброса пакетів;
- перевірка коректності підрахунку і відображення обсягу трафіку;
- перевірка виходу з ПЗ;

- проходження повного циклу користування ПЗ.

В результаті тестування, були усунені знайдені помилки. В результаті повторного тестування додаткових помилок виявлено не було. Але все ж, є ймовірність, що на інших ОС системах, мережевих пристроях, можливі похибки в роботі. Це, так само, може залежати від версій бібліотек, типів мережевих пристроїв і деяких інших чинників.

В цілому, програмне забезпечення в результаті тестування показало стабільну роботу, і готово до експлуатації.

ВИСНОВКИ

В результаті виконання даного проекту було розроблено програмне забезпечення, що дозволяє здійснювати моніторинг мережі, вести облік трафіку, проводити політику розмежування доступу мережевих пристроїв однієї підмережі, до пристроїв іншого, або їх доступу в мережу Інтернет. Програма здійснює підміну адрес, реалізований алгоритм відповіді на ARP пакети. Таким чином, програми виконує функції роутера.

При розробці проекту були детально розглянуті теоретичні аспекти проектування програми. Проведений порівняльний аналіз та обґрунтований вибір мови програмування C#. Обрано середовище розробки Visual Studio 2017 та необхідні бібліотеки для реалізації роботи з мережевими пристроями та мережевим трафіком, а саме SharpPcap.dll, Pcap.NET.dll і Pcap.dll з пакету WinPcap.

У рамках поставленого завдання був створений і налаштований стенд для тестування і налагодження ПЗ, розгорнутий на базі віртуальної машини VMware Workstation 9.0. Головною перевагою використання даного програмного засобу є здійснення контролю над мережевим трафіком, який надходить від користувачів конкретної підмережі. Обмеженням їх доступу в зовнішню мережу, а також можливістю проводити облік трафіку від кожного мережевого пристрою. Тому, хоч програма реалізовувалася в навчальних цілях, практично, її можна застосовувати на реальних мережевих об'єктах, наприклад, мережевими адміністраторами в приватних фірмах, організаціях, для контролю мережевої діяльності їх співробітників. У висновку можна сказати, що реалізоване програмний засіб пройшов етапи тестування і повністю задовольняє вимогам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы. СПб.: Питер, 2010. 882 с.
2. Уилсон Э. Мониторинг и анализ сетей. Методы выявления неисправностей. М.: Лори, 2002. 288 с.
3. Олифер Н. А. Средства анализа и оптимизации локальных сетей. URL: http://citforum.ru/nets/optimize/locnop_07.shtml (дата звернення 13.05.2019)
4. Лавров А. А., Лисс А. Р., Яновский В. В. Мониторинг и администрирование в корпоративных вычислительных сетях. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 60 с.
5. Total Network Monitor 2. Офіційний сайт. URL: https://www.total-network-monitor.ru/ (дата звернення 13.05.2019)
6. Observium. Офіційний сайт. URL: http://www.observium.org/ (дата звернення 13.05.2019)
7. Nagios. Офіційний сайт. URL: https://www.nagios.org/ (дата звернення 13.05.2019)
8. PRTG Network Monitor. Офіційний сайт. URL: https://www.ru.paessler.com/ (дата звернення 13.05.2019)
9. Kismet. Офіційний сайт. URL: https://www.kismetwireless.net/ (дата звернення 13.05.2019)
10. Wireshark . Офіційний сайт. URL: https://www.wireshark.org/ (дата звернення 13.05.2019)
11. Nedi. Офіційний сайт. URL: https://www.nedi.ch/ (дата звернення 13.05.2019)
12. Zabbix. Офіційний сайт. URL: https://www.zabbix.com/ (дата звернення 13.05.2019)
13. Голуб Б.М. C#. Концепція та синтаксис. Львів: Видавничий центр ЛНУ імені Івана Франка, 2006. 136 с.

14. Бібліотека Pcap. Вікіпедія. URL: https://ru.wikipedia.org/wiki/Pcap (дата звернення 13.05.2019)
15. VMware Workstation. Вікіпедія. URL: https://ru.wikipedia.org/wiki/VMware_Workstation (дата звернення 13.05.2019)
16. Кармайкл Э., Хэйвуд Д. Быстрая и качественная разработка программного обеспечения. М.: Издательский дом «Вильямс», 2003. 403 с.
17. Дудзяний І.М. Об’єктно-орієнтоване моделювання програмних систем. Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. 108 с.
18. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер, 2004. 320 с.

Додаток А

Програмний код додатку