Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студенту	*Мішенкову Артему Олександровичу*
	(ПІБ)
академічної групи	*121м-21-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*Інженерія програмного забезпечення*
на тему:	*Розробка та дослідження ефективності впровадження самоорганізованої системи виявлення шкідливого програмного забезпечення на основі методу головних компонент*

А.О. Мішенков

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтин говою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Удовик І.М.*			

Рецензент	*Доц. Кожевніков А.В.*			

Нормоконтролер	*Проф. Лактіонов І.С.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 20 22 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи
магістра

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121М-21-1* *Мішенкову Артему Олександровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження ефективності впровадження самоорганізованої системи виявлення шкідливого програмного забезпечення на основі методу головних компонент*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 31.10.2022 р. № 1200-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт дослідження є процес функціонування самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах.

Предмет дослідження є методи і засоби створення самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах.

Метою роботи є покращення ефективності виявлення аномалій в комп'ютерних системах при використанні самоорганізованих розподілених систем.

Для розв'язання поставлених задач використовувалися методи теорії розподілених систем; теорій множин, графів та штучного інтелекту; головних компонент для виявлення аномалій; теорії комп'ютерних мереж для організації функціонування розподіленої системи.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна одержаних результатів полягає в наступному:

— удосконалено архітектуру розподіленої системи виявлення аномалій в комп'ютерних системах, в якій синтезовано вимоги

самоорганізованості, централізованості, розподіленості, багаторівневості, і на відміну від відомих рішень, дало змогу удосконалити її внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії;

— розроблено новий метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак;

— удосконалено метод виявлення аномалії згідно методу головних компонент в комп'ютерних системах в мережі, який надав змогу застосовувати його не до однієї комп'ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп'ютерних системах в мережі, що на відміну від відомих рішень, при застосуванні надав змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

4 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз існуючих рішень та постановка задачі роботи	12.09.2022-30.09.2022
Аналіз засобів створення програмного забезпечення з реалізації методів і засобів створення самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах	01.10.2022-31.10.2022
Розробка програмного забезпечення та дослідження ефективності запропонованих рішень.	01.11.2022-10.12.2022

Завдання видала _____ *Удовик І.М.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Мішенков А.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *10.09.2022 р.*

Термін подання до ЕК *18.12.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 67 стор., 9 рис., 2 таблиці, 2 додатка, 36 джерел.

Об’єкт дослідження є процес функціонування самоорганізованих розподілених систем виявлення аномалій в комп’ютерних системах.

Предмет дослідження є методи і засоби створення самоорганізованих розподілених систем виявлення аномалій в комп’ютерних системах.

Метою роботи є покращення ефективності виявлення аномалій в комп’ютерних системах при використанні самоорганізованих розподілених систем.

Наукова новизна одержаних результатів полягає в наступному:

1) удосконалено архітектуру розподіленої системи виявлення аномалій в комп’ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості, багаторівневості, і на відміну від відомих рішень, дало змогу удосконалити її внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії;

2) розроблено новий метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп’ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп’ютерних атак;

3) удосконалено метод виявлення аномалії згідно методу головних компонент в комп’ютерних системах в мережі, який надав змогу застосовувати його не до однієї комп’ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп’ютерних системах в мережі, що на відміну від відомих рішень, при застосуванні надав змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

СПИСОК КЛЮЧОВИХ СЛІВ: АНОМАЛІЇ, АРХІТЕКТУРА, МЕТОД, РОЗПОДІЛЕНА СИСТЕМА, ОБСЯГ ДАНИХ

ABSTRACT

Explanatory note: 67 pages, 9 figures, 2 tables, 2 appendices, 36 sources.

The object of research is the process of functioning of self-organized distributed systems for detecting anomalies in computer systems.

The subject of research is methods and means of creating self-organized distributed systems for detecting anomalies in computer systems.

The purpose of the work is to improve the efficiency of detecting anomalies in computer systems when using self-organized distributed systems.

The scientific novelty of the obtained results is as follows:

1) the architecture of the distributed anomaly detection system in computer systems has been improved, which synthesizes the requirements of self-organization, centralization, distribution, multi-level, and, unlike known solutions, made it possible to improve its internal organization of the interaction of the parts of the system center between different levels of the hierarchy and depending on the activity of the system components in a certain time, the basis for which was the distribution of the decision-making center in the system between its components with the division of the center between the upper and lower levels of the hierarchy;

2) a new method of maintaining the integrity of the architecture of a self-organized distributed system in local computer networks was developed, which takes into account the states of system components, transitions between components and determines the next steps of the system, which made it possible to build distributed systems with a single decision-making center, which will become self-organized and can decide on your next steps depending on the effects of malicious software and computer attacks;

3) the method of detecting anomalies according to the method of main components in computer systems in the network was improved, which made it possible to apply it not to one computer station, but to a group of stations in which a self-organized distributed system of detecting anomalies in computer systems is installed in network, which, unlike known solutions, when used made it possible to reduce the volume of data and, accordingly, speed up their exchange between system components.

LIST OF KEYWORDS: ANOMALIES, ARCHITECTURE, METHOD, DISTRIBUTED SYSTEM, VOLUME OF DATA

ЗМІСТ

ВСТУП

Актуальним напрямом, який потребує напрацювання методів і засобів протидії зловмисним діям, є напрям пов'язаний з функціонуванням комп'ютерних мереж, бо вони використовуються практично в усіх підприємствах, організаціях та установах. Проблеми в їх функціонуванні, які викликані впливом зловмисного програмного забезпечення та комп'ютерних атак, а ще гірше, що полягає в приховуванні присутності зловмисника, можуть призвести до фінансових втрат підприємствами, організаціями та установами.

Дослідження зловмисних проявів в корпоративних та локальних мережах можуть бути проведені з використанням апарату математичної статистики. В корпоративних та локальних мережах підприємств, організацій та установ може перебувати велика кількість комп'ютерів і для дослідження процесів, які протікають в них, в тому числі і зловмисних, потрібні ефективні методи та відповідні засоби опрацювання отриманих даних про події. Ефективність протидії зловмисним проявам досягається за рахунок комплексного підходу орієнтованого на інтеграцію методів виявлення та систем, в яких вони реалізовані. Для зловмисників такі підходи суттєво ускладнюють досягнення результативності.

Взагалі поява в комп'ютерах чи комп'ютерних системах і мережах зловмисного програмного забезпечення або комп'ютерних атак, можливо зловмисних дій користувачів, крім безпосереднього виникнення технічних несправностей апаратних пристроїв, вичерпує множину об'єктів, які своєю нестандартною поведінкою можуть привернути до себе увагу. Для обробки подій потрібна розподілена система, яка б збирала та опрацьовувала данні і видавала б результат з дослідження виявлення. Враховуючи потребу опрацьовування даних оперативно і без втручання людини, то така розподілена система повинна бути самоорганізованою.

Розвитком елементів теорії розподілених систем різного призначення активно займаються Бернс Б., Луцький Г. М., Мельник А. О., Марковський Г., Мухін В. Є. Фахівцями, які займаються застосуванням

методів виявлення аномалії є Корченко А. О. [6], Wawryn, K. [4], Mohiuddin Ahmed [10], Xiang Yu [12], Liu, L. [15], Xiaodan Xu [16], Mukrimah Nawir [21].

В роботі пропонується використання самоорганізованих розподілених систем, розроблених згідно принципів централізації та самоорганізації, для виявлення аномалій у комп'ютерних системах.

РОЗДІЛ 1
АНАЛІЗ ВІДОМИХ МЕТОДІВ І ЗАСОБІВ ВИЯВЛЕННЯ АНОМАЛІЙ В КОМП'ЮТЕРНИХ СИСТЕМАХ

1.1 Огляд та поняття виявлення аномалій в комп'ютерних системах

Комп'ютерні системи (КС) продовжують активно використовуватись в усіх сферах діяльності людини. Вони дозволяють суттєво підвищити продуктивність праці та автоматизувати багато складних процесів і це покращує перспективи їх використання в майбутньому.

Програмне забезпечення, яке використовується в КС, виконує дуже важливу роль та забезпечує ефективне вирішення задач. Але із зростанням актуальності задач, при розв'язанні яких використовують КС, зловмисники спрямовують свої зусилля на взяття під контроль КС шляхом впливу на їх програмне забезпечення. Для цього вони застосовують різноманітні атаки на КС [1-4]. Частина з них стає успішною і цим створює проблеми користувачам КС. Для важливих критичних сфер [2] діяльності людини втрата контролю над КС стає серйозною проблемою з відповідними катастрофічними наслідками. Тому, проблемі побудови ефективних систем протидії зловмисним діям спрямованим на КС користувачів все більше приділяють уваги дослідники та розробники відповідних методів і засобів [2-7].

Комп'ютери користувачів, комп'ютерні системи та мережі стали широким полем для проявів зловмисників і залишатимуться ще довго. Для протидії їм важливим є продовження вже напрацьованих досліджень з урахуванням перспектив можливого розвитку зловмисного програмного забезпечення та комп'ютерних атак.

Провідними організаціями, які досліджують і публікують актуальну статистику поширення і створення нового зловмисного програмного забезпечення є Virus Bulleten [8] та AV-TEST [9]. Оприлюднені ними результати досліджень підтверджують постійнезростання кількості зловмисного програмного забезпечення. На рис. 1.1 представлено

динаміку кількості зареєстрованого зловмисного програмного забезпечення на основі даних AV-TEST [9] за 2020 рік.

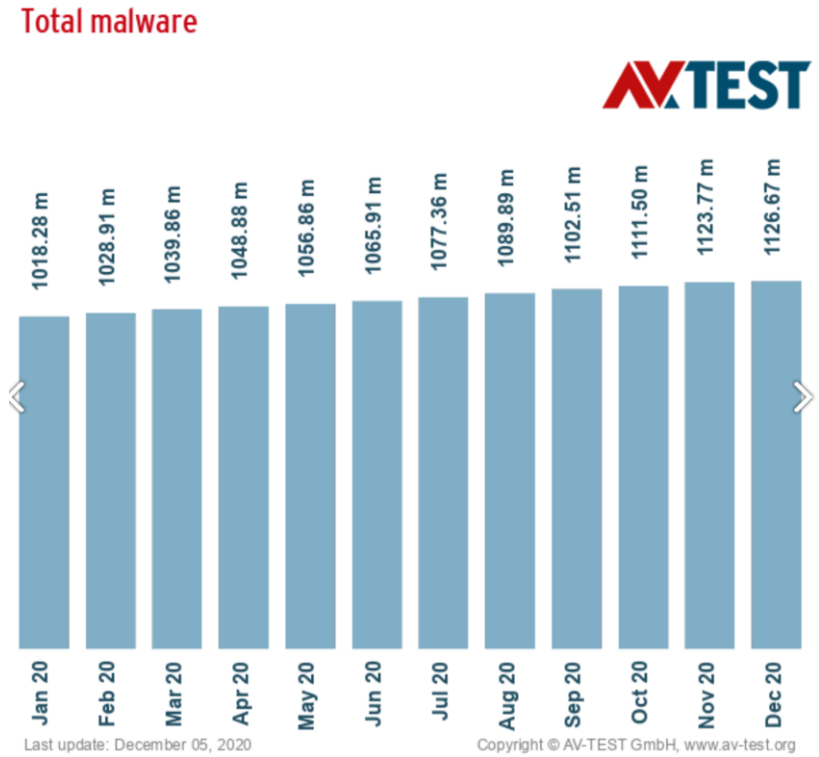

Рис. 1.1. Кількість зареєстрованого зловмисного програмного забезпечення за даними AV-TEST [9]

Одним з найбільш актуальних на сьогодні напрямом, який потребує активного напрацювання методів і засобів протидії зловмисним діям, є напрям пов'язаний з функціонуванням корпоративних та локальних мереж [1, 2], оскільки вони використовуються практично в усіх підприємствах, організаціях та установах. Виведення їх з ладу, а ще гірше, що полягає в приховуванні присутності зловмисника в мережі, може призвести до зупинки роботи підприємства, організації та установи або нанести їм значної фінансової, матеріальної, моральної та організаційної шкоди.

Для дослідження зловмисних проявів в корпоративних та локальних мережах може бути використаний апарат математичної статистики, в якому прояви зловмисної активності будуть відноситись до аномальної поведінки користувачів та досліджуватимуться відповідними статистичними методами. В корпоративних та локальних мережах підприємств, організацій та установ може перебувати велика кількість комп'ютерів і для дослідження процесів, які протікають в них, в тому числі і зловмисних, потрібні ефективні

методи та відповідні засоби опрацювання отриманих даних про події. Ефективність протидії зловмисним проявам досягається за рахунок комплексного підходу орієнтованого на інтеграцію методів виявлення та систем, в яких вони реалізовані. Для зловмисників такі підходи суттєво ускладнюють досягнення результативності.

Якщо розглядати саме комп'ютери в мережах, а не персональні домашні комп'ютери, то відповідно підбір стратегій і методів виявлення, а також і систем виявлення може різнитись. Це пов'язано з можливістю обробки типової інформації, що отримана не з одного комп'ютера, а з декількох. Її інформативність в цьому випадку буде суттєво більшою, ніж у випадку інформації з одного комп'ютера. У випадку одного персонального комп'ютера необхідним є збір статистичних даних певного визначеного набору специфічної інформації протягом тривалого часу, обробка цієї інформації для представлення образу «нормальної» поведінки або її порівняння з підозрілою активністю для формування висновку. У випадку методів і засобів для одного комп'ютера технології виявлення зловмисних дій можуть базуватись на моделі зловмисної поведінки або на порівнянні моделі з потоком подій, які протікатимуть в комп'ютері. Знання про моделі зловмисної поведінки заносяться в базу моделей і можуть бути або статичними сигнатурами або поведінковими сигнатурами. У другому випадку здійснюють формування моделі поведінки та її порівняння з потоком подій, які відбуваються в комп'ютері. Ці ж підходи та стратегії можуть бути використані окремо і для комп'ютерів, які під'єднані в мережу. Але вони можуть, також, комбінуватись з підходами базованими на технологіях виявлення аномальної поведінки. Для комп'ютерів, які під'єднані в мережу, виявлення зловмисної активності на основі дослідження аномальної поведінки може бути ефективніше, ніж підходи базовані тільки на основі моделей зловмисної поведінки. Оскільки, з появою нових моделей зловмисного програмного забезпечення, відомості про які не містяться в засобах виявлення, результат виявлення буде негативним. За період від появи нових моделей зловмисних і до оновлення засобів виявлення зловмисних дій може пройти небагато часу, але таке нове зловмисне

програмне забезпечення може проникнути в комп'ютер або в комп'ютери під'єднані в мережу. Тому, розглянемо технології виявлення зловмисного програмного забезпечення та комп'ютерних атак, які базуються на виявленні аномальної поведінки. Ця технологія використовує знання про нормальну поведінку, яких можна досягти побудувавши профіль користувачів комп'ютерів, під'єднаних до мережі, або ідентифікує аномальні входження в потік подій, які протікають в мережі.

Розглянемо детальніше поняття аномальної поведінки, її особливості з орієнтацією на предметну область та реалізацію в системах певного спеціалізованого типу.

Взагалі поява в комп'ютерах чи комп'ютерних системах і мережах зловмисного програмного забезпечення або комп'ютерних атак, можливо зловмисних дій користувачів, крім безпосередньо виникнення технічних несправностей апаратних пристроїв, вичерпує множину об'єктів, які своєю нестандартною поведінкою можуть привернути до себе увагу. Ці прояви віднесемо до аномальних проявів, виявлення яких досліджуватимемо. Аномалія в перекладі з грецької мови (ανωμαλια) означає ненормальність, неправильність, відхилення від норми, від загальної закономірності. Для формування повної групи подій розглядатимемо також нормальність або нормальну поведінку. Таким чином, процеси в комп'ютерах чи комп'ютерних системах і мережах, які викликані розв'язуванням певних задач класифікуватимемо за нормальною або аномальною поведінкою [3]. Задачі, які розв'язуються, можуть бути задані як користувачами з певною корисною метою автоматизації робіт, так і зловмисниками.

В роботі [4] авторами представлена система виявлення вторгнень, що базується на ідеях з імунної системи людини. Спеціальна штучна імунна система контролює локальну область, що містить критичні файли в операційній системі. Запропонований метод включає сканування файлів та перевірку наявності можливих змін, спричинених зловмисним програмним забезпеченням. Система складається з двох модулів: блок генерації рецепторів, який генерує рецептори, використовуючи оригінальний

метод, базований на шаблонах, і блок виявлення аномалій. Аномалії, виявлені у файлах за допомогою раніше створених рецепторів, повідомляються користувачеві. Впроваджена система та проведено експерименти для порівняння ефективності алгоритмів із ефективністю інших методів генерації рецепторів, який називають методом генерації випадкових рецепторів. У контрольованому середовищі тестування аномалії у вигляді зміненого байта програмного коду вводили в моніторингові програми. Реальні тести цієї системи проводились щодо її продуктивності та масштабованості. Крім цієї роботи, відомо багато ефективних спроб використати штучні імунні системи для виявлення аномалій в різних складових програмної частини КС.

В роботі [5] представлено штучну імунну систему (AIS), яка базується на біологічній імунній системі (BIS). AIS використовується для виявлення зловмисних програм. Найвідомішою моделлю AIS є алгоритм негативного відбору (NSA), і він може використовувати лише звичайні вибірки для тренування. Традиційні алгоритми негативного відбору (АНВ) генерують детектори на етапі навчання, а потім виявляють елементи аномалії на етапі тестування. У традиційних АНВ є певні недоліки. Зокрема, авторами роботи встановлено, що реальні програми часто змінюються, звичайні можуть змінюватися аномальними, і навпаки. Традиційні АНВ легко виробляють багато помилкових тривог та помилкових негативних наслідків у реальних застосунках. Традиційним АНВ бракує здатності до постійного навчання на етапі тестування, і дорого генерувати достатню кількість детекторів, щоб покрити загальний несамостійний простір у навчанні. Щоб подолати недоліки традиційних АНВ, до таких алгоритмів авторами роботи введено нову схему з адаптивним онлайн-навчанням, яка включає в себе побудову відповідного профілю системи, генерування нових детекторів, що закривають отвори несамостійного простору, видаляючи ті детектори, які лежать у самопросторі, зменшують кількість помилкових тривог, а коригування цих детекторів, які частково охоплюють власний простір, зменшує помилкові тривоги та збільшує швидкість виявлення. Таким чином, подальший розвиток застосування відомих методів виявлення аномалій є перспективним в напрямі інтеграції та комбінування

сучасних методів виявлення аномалій.

В авторефератах [6, 7] дисертаційних робіт представлено розроблені методи ідентифікації аномальних станів для систем виявлення вторгнень та стратегії забезпечення функціональної стійкості розподілених інформаційних систем до кібернетичних загроз. Наукові результати цих робіт підтверджують доцільність дослідження аномальних станів для виявлення зловмисного програмного забезпечення та комп'ютерних атак в КС [6], а також застосування засобів, які базуються на використанні розподілених систем для забезпечення функціональної стійкості інформаційних систем [7].

Таким чином, виявлення зловмисного програмного забезпечення та комп'ютерних атак в КС на основі виявлення та встановлення аномальних проявів є перспективним напрямом досліджень. Крім того, застосування різноманітних методів виявлення аномалій до програмного забезпечення та особливо системних програм певного застосування в КС не завжди дозволяє досягти бажаного результату з виявлення зловмисного програмного забезпечення та комп'ютерних атак, тому проведення подальших досліджень з розробки нових підходів до виявлення на основі виявлення аномалій залишається актуальною науковою задачею. Для визначення стратегії її розв'язання першочергово необхідним є проведення аналізу відомих методів виявлення аномалій в КС з метою визначення їх переваг та недоліків, а також виділення нерозв'язаних підзадач, які впливають на досягнення ефективного результату з виявлення. Важливою частиною такого дослідження, яка суттєво впливає на ефективність і особливо достовірність виявлення зловмисного програмного забезпечення та комп'ютерних атак в КС, є засоби, які підтримуватимуть реальне застосування розроблених методів. Засоби, в яких реалізовані методи виявлення суттєво впливають на результат виявлення, тому необхідним є їх розробка інтегровано з імплементованими в них розробленими методами виявлення.

1.2. Відомі методи та засоби виявлення аномалій в комп'ютерних системах

1.2.1. Аналіз відомих методів та алгоритмів виявлення аномалій в комп'ютерних системах

Дослідженню аномалій в комп'ютерних системах для виявлення зловмисного програмного забезпечення і комп'ютерних атак приділяють увагу багато дослідників. Вони розробили багато різних методів виявлення [10-18]. Для розробки нових чи покращення відомих рішень з виявлення аномалій в КС потрібно встановити переваги та недоліків відомих розроблених методів. Виділення нерозв'язаних підзадач, що впливають на досягнення ефективного результату з виявлення, та стратегії з усунення частини недоліків відомих методів дозволять підвищити достовірність та покращити ефективність виявлення. Розглянемо відомі методи виявлення зловмисного програмного забезпечення і комп'ютерних атак в КС, які базуються на встановленні аномальних станів. До розгляду візьмемо ті з них, які запропоновані відомими дослідниками в цій галузі.

У роботі [10] представлений поглиблений аналіз чотирьох основних категорій методів виявлення аномалій, які включають класифікацію, статистику, теорію інформації та кластеризацію. Основна увага зосереджена на проблеми дослідження з наборами даних, що використовуються для виявлення вторгнень у мережу. Результати дослідження дають змогу пов'язати класифікацію, статистичну обробку даних та кластеризацію із поставленою науковою задачею в частині розробки застосовуваних методів при обробці вхідних даних з метою виявлення мережних аномалій.

Проблемі узгодження низьковимірних багатовимірних об'єктів високомірних даних розглянуто в роботі [11] як з теоретичної, так і з обчислювальної точки зору. Оскільки набори даних стають більш неоднорідними та ускладненими, простори, які використовуються для їх апроксимації, повинні стати більш неоднорідними. В цій роботі відображеног результати роботи з переходами до змінених базисів, що є

актуальним в розрізі поставленої задачі з виявлення аномалій і зменшення розмірності даних. Також, проаналізована обчислювальна складність таких перетворень для оцінки необхідних обчислювальних ресурсів.

З розповсюдженням Інтернету речей через бездротові сенсорні мережі генерується величезна кількість даних датчиків з безпрецедентною швидкістю, що призводить до дуже великої кількості явної або неявної інформації [12]. При аналізі таких даних датчиків особливо важливо точно та ефективно виявляти не тільки окремі аномальні поведінки, але й аномальні події (тобто моделі поведінки). Однак більшість попередніх робіт були зосереджені лише на виявленні аномалій, водночас ігноруючи співвідношення між ними. Навіть у підходах, що враховують кореляцію між аномаліями, більшість ігнорує той факт, що аномалія стану даних датчиків змінюється з часом. У цій статті [12] запропоновано безконтрольний метод виявлення аномалій контексту в Інтернеті речей за допомогою бездротових сенсорних мереж, який враховує як статус динамічної аномалії, так і кореляцію між аномаліями, заснованими в контексті на їх просторових та часових сусідах. А, також, досліджено в роботі ефективність запропонованого методу в моделі виявлення аномалії. Внесення в роботу відомостей про аномалії і аномальні прояви є важливим з точки зору врахування динаміки отримання даних.

В роботі [13] процес виявлення несподіваних елементів або подій у наборах даних, які відрізняються від норми, розглядається як пошук аномалій. На відміну від стандартних задач класифікації, виявлення аномалій часто застосовується до немаркованих даних, беручи до уваги лише внутрішню структуру набору даних. Ця проблема відома як неконтрольоване виявлення аномалій і вирішується у багатьох практичних додатках, наприклад, у виявленні вторгнень у мережу, виявленні шахрайства, а також у галузі біології та медицини. У цій статті представлено результати здійсненої оцінки 19 різних алгоритмів виявлення аномалій на 10 різних наборах даних із декількох доменів додатків. Робота є важливою для досліджень безконтрольного виявлення аномалій.

Виявляти та обробляти аномалії для великих даних у режимі реального

часу є складним завданням. Обсяг і швидкість даних у багатьох системах ускладнює типовим алгоритмам масштабування та збереження своїх характеристик у реальному часі [14]. Поширеність даних у поєднанні з проблемою, що багато існуючих алгоритмів враховують лише зміст джерела даних, а не контент. Запропоновані в роботі рішення визначають контекст виявлення аномалій. Він складається з двох різних кроків: виявлення вмісту та виявлення контексту. Детектор вмісту використовується для визначення аномалій у режимі реального часу. Детектор контексту використовується для обрізання результатів детектора вмісту, виявляючи ті аномалії, які вважаються як змістовими, так і контекстно аномальними. Детектор контексту використовує концепцію профілів, які є групами аналогічно згрупованих точок даних, що генеруються багатовимірним алгоритмом кластеризації. Дослідження було оцінено на основі проведених експериментів для двох реальних наборів даних датчиків. Резуьтати цієї роботи [14] важливі в контексті важливості обробки контенту датчиків.

В роботі [15] пропонується поступовий метод безконтрольного виявлення аномалії, який дозволяє швидко аналізувати та обробляти великі дані в режимі реального часу. Оцінка набору даних під час експерименту показує, що метод зближується зі своїм автономним аналогом для нескінченно зростаючих потоків даних.

В роботі [16] проаналізовано відомі рішення з виявлення аномалій, особливо для даних із великими розмірами та змішаними типами, де виявлення аномальних моделей чи поведінки є нетривіальною роботою. В результаті важливість даної роботи в проведених дослідженнях відомих підходів і їх порівнянні, що дозволить врахувати ці результати при виборі перспективних рішень.

В роботі [17] зосереджено увагу на ранньому виявленні несподіваних спостережень у фізичній інфраструктурі, що має велике значення для запобігання поломки системи та подальших втрат. Однак сучасна техніка для виявлення аномалій в існуючій платформі моніторингу інфраструктури головним чином залежить від методу фіксованого порогу.

Очевидним недоліком методу є те, що він, як правило, призводить до високого рівня помилкового виявлення. У цьому дослідженні підхід до виявлення статистичних аномалій запроваджено до моніторингу фізичної інфраструктури. В роботі розглядаються три важливі типи аномалій, які зустрічаються на платформі моніторингу інфраструктури, а саме наївні точкові аномалії, контекстні аномалії точок та зміщення рівня. В роботі пропонується до застосування розроблений метод, заснований на моделі Гаусса, для виявлення зазначених трьох аномалій. Оскільки запропонований метод може ефективно виявляти лише наївні точкові аномалії; запропоновано вдосконалений підхід, що поєднує результати статистичних випробувань на вихідних та першовідмінних даних моніторингу. Оцінюються результати запропонованих методів на реальному наборі даних. Результати показують, що оптимізований підхід до виявлення аномалій має хорошу точність і може значно знизити швидкість неправильного виявлення. Отримані результати дають розуміння обробки трьох типів аномалій.

Однією із сучасних проблем виявлення аномалій є здатність виявляти і розрізняти як точкові, так і колективні аномалії в межах послідовності даних або часових рядів [18]. В роботі [18] розроблено метод та засоби, щоб надати користувачам вибір методів виявлення аномалій, і, зокрема, забезпечує реалізацію нещодавно запропонованого сімейства алгоритмів виявлення аномалій. У статті [18] описуються реалізовані методи, а також висвітлюється їх застосування до модельованих даних, а також реальні приклади даних, що містяться в пакеті. Поділ на точкові і колективні аномалії, а також, методи їх виявлення є важливим в розвитку методології з виявлення аномалій.

Метою роботи [19] є швидке та точне виявлення ненормальних даних складного та складного промислового обладнання із датчиками. Завдяки стрімкому розвитку Інтернету речей, все більше обладнання обладнується датчиками, особливо більш складне та складне промислове обладнання встановлюється з великою кількістю датчиків. Для моніторингу роботи обладнання швидко збирається велика кількість даних моніторингу. Обробка таких даних, причому у великій кількості, представлена в роботі.

В роботі [20] представлено застосування такого методу, який називається однокласною машиною векторної підтримки, для пошуку аномальних шаблонів серед джерел, попередньо вибраних із середньо-інфрачервоного каталогу. Для створення моделі очікуваних даних в роботі описано результат тренувань алгоритму на наборі об'єктів зі спектроскопічними ідентифікаціями. Виявлення аномалій додає гнучкості автоматизованим процедурам поділу джерел та допомагає перевірити надійність та репрезентативність навчальних зразків. Таким чином, це слід розглядати як важливий крок у контрольованих схемах класифікації для забезпечення повноти та чистоти створених каталогів.

У роботі [21] описано загальний механізм аналітики, який забезпечує надійні попередження про зміни та аномалії в сенсорному потоці даних. У той час як більшість існуючих аналітичних реалізацій IoT вимагають припущень, що стосуються конкретних областей, в роботі надано значну інформацію через методи машинного навчання та вдосконалені статистичні тести без попередніх знань. Система виявлення аномалій мережі дозволяє контролювати комп'ютерну мережу, яка поводиться інакше, ніж мережевий протокол, і її багато застосовується в різних доменах. Проте, проблема виникає там, де різні домени застосунків мають різні визначальні аномалії у своєму середовищі. Вони ускладнюють вибір найкращих алгоритмів, які відповідають вимогам певних доменів. Крім того, проблема централізації, яка спричиняє руйнування мережевої системи, коли в систему вливається потужний зловмисний код. Тому в цій роботі показано результати проведеного експерименту із використанням контрольованого машинного навчання для системи виявлення аномалій мережі, яка мінімізує вартість зв'язку та пропускну здатність мережі, мінімізовану за допомогою набору даних для порівняння їх продуктивності в термінах їх точності та часу обробки для класифікатора для побудови моделі. В результаті, розподілений алгоритм вирішує проблему централізації з точністю та часом обробки, як і раніше, значним у порівнянні з централізованим алгоритмом, хоча є певна втрата точності та часу.

1.2.2. Аналіз методів та стратегій створення розподілених систем

Побудова розподілених систем в локальних комп'ютерних мережах поряд з розробкою нових методів чи удосконаленням відомих з виявлення аномалій є актуальною, бо від ефективності їх функціонування залежить оперативність при виявленні аномалії та реагування на неї. Крім того, ефективна інтеграція при імплементації методу виявлення аномалії в розподілену систему може покращити загальну ефективність з виявлення аномалії та реагування на неї. Розглянемо відомі методи, що стосуються проєктування розподілених систем та оптимізаційні стратегії для покращення їх ефективності функціонування.

Для розподілених систем вартість зв'язку є найбільш часто використовуваною метрикою з метою оцінки ефективності операцій у розподілених алгоритмах для середовищ передачі повідомлень [22]. При цьому постійне припущення полягає в тому, що вартість обчислень в компонентах незначна порівняно з вартістю зв'язку. Однак у багатьох випадках реалізації операцій покладаються на складні обчислення, які не слід ігнорувати. Тому більш точна оцінка ефективності роботи повинна враховувати як обчислювальні витрати, так і витрати на зв'язок. У роботі [22] основна увага приділяється ефективності операцій читання та запису в емуляціях атомної спільної пам'яті читання / запис в асинхронному середовищі, що передає повідомлення, схильному до збоїв. В роботі розроблено і запропоновано новий обчислюваний предикат та алгоритм його обчислення за лінійний час. Результати опубліковані в роботі [22] надають нового значення терміну швидкості, оцінюючи як зв'язок, так і ефективність обчислень кожної операції і є важливими для побудови розподілених систем в частині організації зв'язку між компонентами та обчисленнями в них.

В роботі [23] проаналізована ефективність слабо узгоджених, але швидко реагуючих розподілених сховищ даних. Актуальність цього напряму обгрунтована проблемами, які пов'язані з узгодженістю між розробкою

складних застосунків та отриманням лише слабких гарантій узгодженості в сховищах даних. Компроміс узгодженості спрямований на досягнення як сильної узгодженості, так і низької затримки у загальному випадку. У розподілених системах зберігання досліджене в [23] загальне поняття майже сильної узгодженості з точки зору розробки алгоритмів швидкого зчитування, одночасно гарантуючи ймовірнісну атомність з чітко обмеженою кількістю записів одночасно. Загальний випадок цієї проблеми полягає в тому, коли кілька клієнтів можуть писати дані в розподілених сховищах даних. Важливим показником в цьому випадку є межа застарілості даних та ймовірність порушення атомарності, розкладаючи непослідовні зчитування на інверсію читання та шаблони інверсії запису. Результат цієї роботи важливий для організації розподілених сховищ даних і узгодження інформації в них за критерієм актуальності.

В роботі [24] представлена розподілена система для управління дуже великими обсягами структурованих даних, розподілених на багатьох товарних серверах, забезпечуючи при цьому високодоступні послуги без жодної точки відмови. Ця система може працювати на інфраструктурі з сотнями вузлів розповсюджено за різними центрами обробки даних. Вона не підтримує повну реляційну модель даних, а надає клієнтам модель простих даних, яка підтримує динамічний контроль над розміщенням даних та форматуванням.

В роботі [25] представлено створені оптимальні комунікаційні протоколи у трьох сценаріях, що є важливим для підтримки цілісності розподіленої системи. Проблема складності при організації комунікації пошуку наближеного максимального узгодження в багатосторонній моделі передачі повідомлень є актуальною [26]. Задача максимального узгодження є однією з найбільш фундаментальних комбінаторних задач графа, що має різноманітні програми, і вирішенню оцінки її складності присвячена робота авторів в [26]. Проблемам виявлення мережевих структур та їх топологій присвячена робота [27], яка відіграє центральну роль у розподілених обчисленнях. В роботі [28] досліджено різні налаштування розподілених обчислень та відповідні для них методи. У роботі [29]

досліджено обчислювальну потужність популяційних протоколів за деяких ненадійних або слабших моделей взаємодії. Це необхідно для організації підтримки цілісності розподілених систем.

Протягом багатьох років елегантна ієрархія консенсусу, що базується на обчислюваності, була найкращим поясненням відносної сили різних об’єктів [30]. Оскільки справжні мультипроцесори дозволяють застосовувати різні інструкції, які вони підтримують, до будь-якого місця пам'яті, то можливість поєднання інструкцій, що підтримуються різними об’єктами, а не розгляд колекції різних об’єктів є актуальним. У цій роботі запропоновано класифікацію відносної потужності наборів багатопроцесорних команд синхронізації на основі складності, що охоплюється мінімальною кількістю місць пам'яті необмеженого розміру, необхідних для вирішення безперешкодного консенсусу при використанні різних наборів інструкції [30]. Вивченню складності повідомлення неявних виборів лідера в синхронних розподілених мережах діаметром два присвячені дослідження в роботі [31]. Результати характеризують складність повідомлення виборів лідера щодо діаметра графіка [31]. В роботі [32] розглянуто проблеми планування в моделі потоку даних розподіленої транзакційної пам'яті. Об'єкти, спільно використовувані транзакціями, переміщуються з одного мережного вузла на інший, слідуючи мережним шляхам. Авторами досліджено, як передача об’єктів у мережі впливає на час завершення всіх транзакцій та загальну вартість зв'язку. Розроблені авторами алгоритми планування, які для зв’язку працюють майже оптимально і ефективно виконуються за часом. У роботі [33] розглядається модель зв'язку, в якій в кожному раунді кожен агент витягує інформацію з кількох випадково обраних агентів. Таким чином, метою є визначення найменшлшл обсяг інформації, виявленої в кожній взаємодії (розмір повідомлення), що, тим не менше, дозволяє ефективно та надійно обчислювати основні завдання розповсюдження інформації. Розроблений протокол використовує лише 3 біти на взаємодію.

У роботі [34] пропонується нова техніка розподіленого за функціями зловмисного програмного забезпечення, яке динамічно

розподіляє свої функції серед багатьох програмних компонентів, щоб обійти різні механізми безпеки, такі як додавання до білого списку програм та виявлення поведінки антивіруса. Для оцінки такого підходу авторами впроваджено інструмент, який автоматично генерує такі екземпляри зловмисного програмного забезпечення, та проведено серію експериментів, що показують ризики.

Розпізнати цільове зловмисне програмне забезпечення за допомогою антивірусів, IDS, IPS та спеціальних засобів виявлення досить складно [35]. Автори порівнюють різні методи машинного навчання, що використовуються для аналізу шкідливих програм, зосереджуючись на статичному аналізі.

В роботі [36] авторами розробляємо офіційну систему пасивного тестування програмних систем, де сторони спілкуються асинхронно.

Авторами в роботі [37] розроблено протокол, який з великою ймовірністю підраховує розмір розподіленої системи, компоненти якої з самого початку є добре змішаними і в процесі функціонування кооперуються. В роботах [38-41] авторами, також, досліджуються протоколи для розподілених систем.

Робота [42] авторів стосується локальних проблем розподілених обчислень та досліджує розрив між рандомізованими та детермінованими рішеннями за обмеження пропускної здатності. Їх головний внесок полягає у наданні інструментів для дерандомізації рішень локальних проблем. В роботах [43-45] досліджується ця ж проблема і запропоновано її рішення.

В роботі [46] авторами досліджено процеси, які обмінюються даними через спільну пам’ять в розподілених системах, з метою встановлення можливості випадковим чином отримувати їх з основного планувальника. Ними представлено загальний метод обчислення цих величин шляхом класифікації розподілених алгоритмів за їхньою схемою доступу до спільної пам'яті. В роботах [47-49] досліджується цю ж проблема, яка пов’язана з генерацією випадкових чисел.

В роботі [50] авторами представлено підхід для керування

ресурсами обчислювальної мережі за умови встановлення довіри до компонентів системи.

Таким чином, розробка розподіленої системи виявлення аномалії в комп'ютерних системах базується на двох складових: розробка розподіленої системи; використання та удосконалення методів виявлення аномалії. Аналіз наукових результатів з цих напрямків є основою для створення розподіленої системи виявлення аномалії в комп'ютерних системах.

1.2.3. Аналіз існуючих систем з виявлення аномалій та вторгнень в комп'ютерні системи

Найближчими програмними рішеннями до розроблюваної системи виявлення аномалій в локальних комп'ютерних мережах згідно поставленої задачі є мережні системи виявлення вторгнень, мережні антивірусні програми та некомерційні розробки відповідного призначення.

Мережна система виявлення зловмисного програмного забезпечення або комп'ютерних атак переважно має модуль централізованого управління [51-54]. Це дає змогу адміністратору системи керувати оновленнями і налаштуваннями всіх параметрів в мережі з єдиної консолі. Мережні антивірусні засоби, як правило, використовують сумісно із засобами антивірусного захисту вузлів мережі в якості другого рівня захисту [55]. За такої архітектури мережних систем рекомендованим є використання мережних і хостових частин від різних виробників. Наприклад, в якості технології захисту у мережних системах використовується система захисту хостів у корпоративних мережах [52, 56]. Її елементи дозволяють здійснювати контроль над застосунками, веб-трафіком і пристроями, які під'єднуються. Керування функціями системи здійснюється з єдиної консолі. Мережною системою від компанії «Dr.Web» [57, 58] є застосунок «Dr.Web CureNet!». Він побудований згідно централізованої архітектури. Symantec Endpoint Protection є мережною антивірусною системою. Вона забезпечує адміністратора мережі необхідними інструментами з розгортання антивірусних засобів в мережі [53]. Апаратно-програмна система для

виявлення зловмисного програмного забезпечення та комп’ютерних атак розроблена компанією Palo Alto Networks [59, 60]. Мережна система «Malwarebytes Endpoint Security» [61] надає розширений набір локальних засобів для виявлення і видалення загроз в комп'ютерах у мережі. Технологія «Cisco® Network Admission Control (NAC)» була розроблена для забезпечення захисту усіх хостів в мережі [57, 58]. Мережна система Kaspersky Administration Kit реалізує самостійну роботу без втручання на етапі дослідження чи виявлення адміністратора [62].

Засоби для хостових систем, які призначені для виявлення зловмисного програмного забезпечення або комп’ютерних атак: Avast! [63], AVG Antivirus [64], AntiVir (Avira) [65], BitDefender [66], Clam AntiVirus [60] тощо.

Таким чином, результати аналізу, зокрема в [8, 9], показують, що хостові засоби та мережні системи виявлення зловмисного програмного забезпечення або комп’ютерних атак не забезпечують його повного виявлення. З такими результатами погоджуються і виробники засобів і користувачі цих засобів. Перспективним для покращення достовірності виявлення є розробка і використання мережних системи виявлення, які б базувались на виявленні аномалій у вузлах в мережі.

1.3. Постановка задачі дослідження

Для розв’язання наукової задачі з покращення ефективності виявлення аномалії в комп’ютерних системах, прояви якої зумовлені впливами зловмисного програмного забезпечення та комп’ютерних атак необхідно здійснити розроблення методів та розподілених систем з виявлення аномалій на основі синтезу в них принципів самоорганізації та централізації і розв’язати такі завдання:

1) дослідити особливості прояву аномалії в комп’ютерних системах за умов функціонування зловмисного програмного забезпечення і здійснення комп’ютерних атак в локальних комп’ютерних мережах та проаналізувати сучасні методи виявлення аномалії, їх особливості та методи створення і архітектури розподілених систем;

2) удосконалити модель архітектури розподіленої системи виявлення аномалії в комп’ютерних системах, в якій синтезувати вимоги розподіленості, централізованості та самоорганізованості, для створення на її основі розподілених систем та їх компонентів, що функціонуватимуть під керівництвом одного центру розподіленого між різними компонентами і самостійно прийматимуть рішення про наявність аномалії;

3) розробити метод підтримки цілісності самоорганізованої розподіленої системи виявлення аномалії в комп’ютерних системах для підтримки її цілісності, на основі якого система змогла б самостійно змінювати свою архітектуру без втручання користувача, а також визначати стратегію своєї подальшої роботи;

4) удосконалити метод централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент для зменшення розмірності з моменту отримання та надсилання даних в центр прийняття рішень системи;

5) розробити програмне забезпечення самоорганізованої розподіленої системи виявлення аномалії в комп’ютерних системах для підтвердження можливості практичного створення таких систем згідно запропонованих результатів роботи та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення.

РОЗДІЛ 2
АРХІТЕКТУРА САМООРГАНІЗОВАНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛІЙ В КОМП'ЮТЕРНИХ СИСТЕМАХ НА ОСНОВІ МЕТОДУ ГОЛОВНИХ КОМПОНЕНТ

2.1 Архітектура самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах

Виявлення аномалій в комп'ютерних системах, які викликані проявами зловмисного програмного забезпечення або в результаті здійснення комп'ютерних атак потребують не тільки ефективних методів, які з високим ступенем достовірності встановлюватимуть наявність зловмисних аномальних проявів, але не менш важливим є система, в яку імплементовано розроблені методи.

Вимоги до системи, в яку будуть імплементовані розроблені, потрібно задавати такі, щоб в подальшій своїй роботі така система могла підтримувати свою працездатність в умовах проявів зловмисного програмного забезпечення або в результаті здійснення комп'ютерних атак. Якщо система в умовах зловмисних проявів не зможе підтримувати свою працездатність, тоді і методи, які закладені в неї з виявлення аномалій, не будуть застосовними. Тому, вимоги до системи такого призначення повинні бути сформовані з врахуванням не тільки особливостей застосування, але і враховуючи середовище функціонування, в якому, наприклад, здійснюватимуться зловмисні прояви.

Система виявлення аномалій в комп'ютерних системах для реалізації можливості залучення інформації з різних комп'ютерних станцій, під'єднаних в локальну мережу, повинна мати розподілену архітектуру, оскільки в такому випадку вона зможе скористатись перевагами залучення більшої обчислювальної потужності за рахунок об'єднання обчислювальних ресурсів всіх комп'ютерних станцій, в яких встановлені її компоненти, порівняно з зловмисними проявами, які можуть надходити чи здійснюватись в одному або декількох вузлах в мережі. Розподіленість в її архітектурі в

мережі в комп'ютерних станціях надає переваги над зловмисним програмним забезпеченням або при здійсненні комп'ютерних атак на комп'ютерні системи в мережі завдяки можливості першочергово забезпечити функціонування компонентів у вузлах мережі, які не піддаються атаці чи впливу зловмисного програмного забезпечення, і тому можуть бути залучені до процесу виявлення, навіть за умови втрати частини системи через виведення її компонентів з безпечного стану через втрату контролю над вузлами в мережі через їх ураження зловмисним програмним забезпеченням або внаслідок успішно проведеної комп'ютерної атаки. Але розподілення компонентів системи між різними комп'ютерними станціями в архітектурі системи має недолік, який пов'язаний першочергово з витратами часу на пересилання зібраних для обробки даних до центру чи центрів прийняття рішень, а потім так само поверненням результатів обробки в формі прийнятого рішення щодо подальших дій компонентів системи. Побудова відповідної архітектури системи, яка б враховувала баланс переваг і недоліків такої архітектури та вплив на її компоненти проявів зловмисного програмного забезпечення чи комп'ютерних атак, є актуальною науковою задачею. Для швидкої взаємодії компонентів системи важливим є метод організації взаємодії компонентів системи, а також оптимізація зв'язків між частинами системи в процесі оперативного реагування на аномальні прояви. Якщо б події відбувались в межах однієї комп'ютерної станції, тоді б і рішення про наявність аномальних проявів приймалось безпосередньо в ній згідно певного реалізованого методу в системі, яка розміщена саме в комп'ютерній станції і з іншими вузлами в мережі не має комунікації в питаннях, які стосуються виявлення аномалій. Але в такому хостовому випадку не має гарантії того, що час обробки подій, встановлення факту наявності аномальних проявів буде суттєво меншим або взагалі менше, ніж час витрачений на комунікацію і пересилання повідомлень між компонентами розподіленої системи за умови, що ефективність методів, які імплементовані в неї, та взаємодії між її компонентами можуть бути швидшими при обробці подій, пов'язаних з аномальними проявами. Таким чином, виявлення аномалій окремими хостовими системами в окремих комп'ютерних станціях

порівняно з виявленням розподіленими системами в багатьох вузлах в мережі за витраченим часом може бути різним, зокрема, як більшим, так і меншим. Крім того, хостова система в комп'ютерній станції може не впоратись із зловмисними проявами самостійно і її робота з виявлення зловмисних проявів може суттєво затягнутись в часі. В зв'язку з такими проблемами, які виникають при вирішення питання про архітектуру систем виявлення аномалій та її місце розміщення та місце використання, сформулюємо наступні гіпотези. До першої групи гіпотез віднесемо врахування застосування і розміщення для хостових або мережних систем. Тоді, нехай гіпотеза H_0 – розміщення та застосування хостових систем виявлення аномальних проявів ефективніше порівняно із застосуванням мережних систем. Гіпотеза H_1 – розміщення та застосування хостових систем виявлення аномальних проявів неефективне порівняно з мережними системами. Класифікуємо результати за цими двома гіпотезами за критерієм отримання кращого результату і представимо в таблиці спряження на основі результатів класифікації і фактичної належності класам.

Таблиця 2.1 – Таблиця спряження

Гіпотеза	Результати правильно класифікованих об'єктів для хостових систем, %	Результати неправильно класифікованих об'єктів для хостових систем, %
H_0	K_{TP}	K_{FP}
H_1	K_{FN}	K_{TN}

Позначення в таблиці 2.1 означають такі величини:

1) $K_{TP}\%$ - відсоток правильно класифікованих об'єктів для хостових систем; 2) $K_{FP}\%$ - відсоток неправильно класифікованих об'єктів для хостових систем як правильно класифікованих об'єктів (відсоток помилок другого роду; відсоток хибного виявлення);

3) $K_{FN}\%$ - відсоток дійсних об'єктів для хостових систем класифікованих неправильно (помилка першого роду; відсоток хибно пропущених об'єктів);

4) $K_{TN}\%$ - відсоток правильно класифікованих неправильних об’єктів об’єктів для хостових систем.

За результатами тестування, яке проводиться на постійній основі регулярно, антивірусних засобів та систем виявлення вторгнень провідними загальновизнаними лабораторіями [1, 2] стосовно достовірності виявлення ними зловмисного програмного забезпечення та комп’ютерних атак було здійснено вибірку результатів тестування хостовими та мережними системами і розподілено їх два класи. Данні, які отримані в результаті їх розподілу, оброблені за гіпотезами та представлені відповідно між чотирма класами. В результаті такого дослідження було встановлено, що достовірність виявлення зловмисного програмного забезпечення та комп’ютерних атак антивірусними засобами та системами виявлення вторгнень усереднена за двома типами засобів для виявлення в хостових засобах хостові засоби мають менший відсоток виявлення в комп’ютерній станції порівняно з виявленням мережними засобами в окремій комп’ютерній станції. Але в такому випадку мережні засоби використовуються не тільки для здійснення виявлення в окремій комп’ютерній станції, але мають ще спрямованість на інші вузли в комп’ютерній мережі. Така спрямованість мережних систем виявлення призводить до витрат часу на задачі з виявлення, як в окремих комп’ютерних станціях, так і в цілому в мережі. Все це вимагає відповідних витрат часу, які можуть перевищити витрати часу на виконання завдань з виявлення окремими хостовими системами в окремих комп’ютерних станціях, причому цей процес здійснюється паралельно. Тому, фактично наявними є такі варіанти архітектури системи виявлення аномальних проявів: хостова система; мережна система з основними задачами виявлення в мережі та її вузлах; мережна система винятково для виявлення аномальних проявів в мережі; мережна система для виявлення аномальних проявів в мережі та в кожній комп’ютерній станції під’єднаній в мережу. Системи виявлення вторгнень як окремі системи і комбіновані з системами виявлення аномалій не розглядатимемо. Таким чином, виберемо тип мережної системи з розглянутих та проаналізований такий, що включатиме в себе необхідність вирішення задач виявлення аномалій

в мережі та в кожній комп'ютерній станції під'єднаній в мережу. Вибір такого типу системи виявлення аномалій на основі дослідження проведеного з джерел [1, 2] надасть можливість здійснювати виявлення аномалій і засобами хостової частини системи з повноцінним функціоналом та залученням мережної частини системи, яка надаватиме додаткові обчислювальні потужності та імплементовані в неї методи. Такий вибір спрямування системи виявлення аномалій на вузли в мережі та саму мережу визначатиме відповідну її архітектуру, особливістю якої буде розподілення в мережі. Місцем розміщення такої системи виберемо локальну мережу. Масштабування розподіленої системи виявлення аномалій в корпоративній мережі за потреби можна буде здійснити в межах окремих її сегментів локальних мереж не тільки незалежно одна від однієї, а й інтегровано. Необхідність локалізації місця застосування розподіленої системи обгрунтовано ще тим, що відомості про вузли мережі відомі адміністратору і можуть бути враховані в архітектурі розподіленої системи при налаштуванні її під час встановлення. Локалізація місця встановлення розподіленої системи дає змогу отримати перевагу над зловмисними проявами в окремих комп'ютерних станціях не тільки за рахунок залучення більшої потужності обчислювальних ресурсів, але й за рахунок прийняття рішення про наявність аномалії не безпосередньо у атакованій комп'ютерній станції, а в окремому центрі, що суттєво підвищує довіру до отриманого результату.

Прийняття рішень системою виявлення аномалій повинно здійснюватись або в одному центрі або в розподілених центрах на різних рівнях ієрархії. Якщо прийняття рішень буде відбуватись тільки винятково в одному центрі, тоді вся інформація повинна пересилатись в нього, очікувати на обробку, прийняття рішення і надсилання його іншим компонентам системи для виконання подальших дій. Все це може суттєво сповільнити роботу системи в цілому, якщо в центрі накопичиться багато завдань з різних компонентів системи, і може призвести до втрати актуальності прийнятого рішення, бо процеси в мережі та окремих її вузлах виконуються швидко і тому потребують динамічного реагування на події. Визначення місця прийняття

рішень з питань, які пов’язані з функціонуванням системи або з результатами обробки подій у мережі та її вузлах з використанням імплементованих в систему методів, потрібно розподілити в залежності від їх важливості та віднесення до частини системи чи до всієї системи. Найкращим рішенням в такому випадку було б рішення, коли центр прийняття рішень був би найближче до тієї частини системи, яка його потребує. В такому випадку центр прийняття рішень повинен розподілитись між рівнями в компонентах системи. Для реалізації цього необхідно в архітектурі системи вибудувати ієрархічні рівні. Хоча компоненти різних рівнів можуть комунікувати між собою, але на кожному рівні ієрархії будуть центри прийняття рішень, які матимуть можливість приймати рішення тільки з певних чітко визначених питань згідно отриманих з компонентів системи зібраних початкових даних. Але не завжди чітко визначені функції, наприклад, реагування на встановлені аномальні прояви, можуть бути віднесені тільки до центру прийняття рішень, що знаходиться в компоненті нижнього рівня ієрархії. Такі реагування на аномальні прояви повинні бути узгоджені або з самого початку або після первинної реакції основним центром прийняття рішень всієї системи. Також, інші центри прийняття рішень, також, повинні оперативно інформуватись про встановлення аномальних проявів у певному вузлі мережі. Таким чином, правильний розподіл в системі центрів прийняття рішень, визначатиме її ефективність і можливість оперативного реагування на виявлені аномальні прояви.

Самоорганізованість як характерна особливість проєктованої системи виявлення аномалій є необхідною, оскільки події в комп’ютерних системах відбуваються дуже швидко і реагування на них повинно бути таким, щоб результат обробки та прийняття рішення був актуальним, а не із запізненням. Хоча він може надходити із запізненням і бути врахованим, але переважно, враховуючи наслідки впливу зловмисного програмного забезпечення та комп’ютерних атак, є необхідним швидке реагування на події. Якщо б пропонований результат обробки подій з метою виявлення аномалій для остаточного прийняття рішення в кожній ситуації яка виникає у вузлах мережі, в які встановлено компоненти системи, покладався

винятково на системного адміністратора мережі або фахівця з кібербезпеки, тоді більшість подій оброблялись би із суттєвим запізненням. Таке залучення до прийняття рішень системного адміністратора мережі або фахівця з кібербезпеки може бути потрібним на етапі аналізу журналу реєстрації виявлених аномальних подій, зареєстрованим системою і прийнятих нею рішень. Але оперативність в прийнятті рішень найкраще покласти саме на систему, тому в її основі має бути можливість до самоорганізованості для визначення своїх подальших кроків. В загальному така характеристична властивість системи як самоорганізованість може містити механізми не тільки визначення подальших наступних кроків, але і механізми з динамічної перебудови своєї архітектури в залежності від впливів зловмисного програмного забезпечення, комп'ютерних атак, а також результатів оброблених подій з виявлення аномалій. Самоорганізованість системи на рівні закладених в систему механізмів і функцій може бути реалізована як частина основного центру прийняття рішень, тобто тієї частини центру, яка знаходиться на верхньому рівні ієрархії.

Враховуючи такі характерні властивості та особливості проєктованої системи як самоорганізованість та розподіленість, потребує вирішення питання динамічного формування системи з наявних активних компонентів на певний час. Це повинно бути відповідним чином спроєктовано, як на випадок для початкового формування, так і у випадку тривалого використання системи та зміни її архітектури в залежності від зміни активних її компонентів та реагування на аномальні прояви.

Специфіка застосування проєктованої системи виявлення аномалій в комп'ютерних системах пов'язана з руйнуючими впливами зловмисного програмного забезпечення та комп'ютерних атак, причому ці руйнуючі впливи можуть стосуватись і функціонування проєктованої системи з метою виведення її з ладу або окремих її компонентів чи спотворення передаваних даних між компонентами системи. Складність виявлення зі сторони зловмисника систем захисту комп'ютерних систем полягає, зокрема і у відсутності відомостей про засоби захисту атакованих систем. Це дозволяє здійснювати ефективний захист та виявлення аномальних

проявів такими системами без втрати відповідної функційності з виявлення аномалій чи частини функційності. За умови проєктування системи, як розподіленої у вузлах мережі, досить важливою стає організація правильної взаємодії компонентів проєктованої системи на основі нового мережного протоколу передачі даних або удосконаленням існуючого, але який би враховував специфіку задач і ускладнених зловмисним програмним забезпеченням чи комп'ютерними атаками умов, в яких здійснюватиметься обмін інформацією між компонентами системи. Протокол, який регламентуватиме обмін повідомленнями між компонентами проєктованої системи, повинен мати додаткові елементи для підтвердження отримання повідомлення, враховувати динамічне формування системи з її компонентів в різний час, уникнення блокування компоненту системи у випадку затримки повідомлення тощо. Тобто вимоги до протоколу обміну інформацією між компонентами системи повинні бути інші, ніж при відомих, наприклад, IRC. Така вимога до протоколу потрібна, також, для того, щоб підтримувати цілісність проєктованої системи.

Архітектура проєктованої системи в процесі її функціонування повинна динамічно сформуватись з обов'язкової компоненти, в якій частина центру верхнього рівня ієрархії, та частини компонентів системи, необов'язково всіх компонентів. Не обов'язково, щоб усі компоненти проєктованої системи в процесі її функціонування, були наявними та активними. Частина з них може бути у вимкнених комп'ютерних станціях або в процесі функціонування системи частину комп'ютерних станцій з її компонентами користувачі вимкнуть. В такому випадку система повинна продовжувати виконання своїх функцій.

Функціонування хостових компонентів системи окремо за відсутності центру системи і виконання ними повноцінних дій з виявлення аномальних проявів засобами імплементованих функцій повинно підтримуватись, оскільки центр тимчасово може бути недоступним і, тоді, вся система не зможе протистояти зловмисному програмному забезпеченню і комп'ютерним атакам. Важливою є реалізація за таких випадків горизонтального інтерфейсу між компонентами системи нижнього

ієрархічного рівня. Це надало б змогу ефективніше протистояти зловмисним проявам за відсутності центру. Для організації такої взаємодії компонентів необхідним є відповідний протокол і обробка таких подій частинами центру, що знаходиться в компонентах проєктованої системи.

При таких вимогах до архітектури проєктованої розподіленої системи, в якій хостові частини системи розміщені в комп'ютерних станціях повинні приймати рішення про наявність аномалії та передавати результат виявлення в центр розподіленої системи і мережна частина повинна виконувати завдання з виявлення аномалій в локальній мережі, в ній необхідно синтезувати наступні характерні властивості, методи та функційні можливості:

1) централізованість (єдиний центр прийняття рішень) у прийнятті рішень в системі;

2) наявність рівнів ієрархії в питанні прийняття рішень в розподілених частинах центру в усіх компонентах системи у вузлах мережі;

3) розподілення компонентів системи у різних вузлах в мережі;

4) самоорганізованість системи при прийнятті рішень про подальші кроки в роботі системи та її компонентів;

5) динамічне формування системи в процесі її функціонування, як під час початкового формування, так і в процесі тривалого використання;

6) мережний протокол для взаємодії компонентів проєктованої системи;

7) динамічне формування архітектури системи в процесі її функціонування з обов'язкової компоненти, в якій частина центру верхнього рівня ієрархії, та частини компонентів системи, необов'язково всіх компонентів;

8) функціонування хостових компонентів системи окремо за відсутності центру системи і виконання ними повноцінних дій з виявлення аномальних проявів засобами імплементованих функцій;

9) імплементація методів виявлення аномалій в комп'ютерних системах в проєктовану систему.

Синтезуючи виділені характерні властивості, методи та функційні можливості, отримуємо самоорганізовану розподілену систему, яка здатна

функціонувати в локальній комп'ютерній мережі і вирішувати завдання з виявлення аномалій в комп'ютерних системах при наповненні її відповідним функціоналом. Представимо самоорганізовану розподілену систему як множину її компонентів.

Представимо синтезовану архітектуру самоорганізованої розподіленої системи з врахуванням її подання через множину компонентів та характерних властивостей, методів виявлення аномалій, функційних можливостей структурною схемою, яку зображено на рис. 2.1. Архітектура самоорганізованої розподіленої системи з відображенням центру системи в якості компоненти зображена на рис. 2.2.

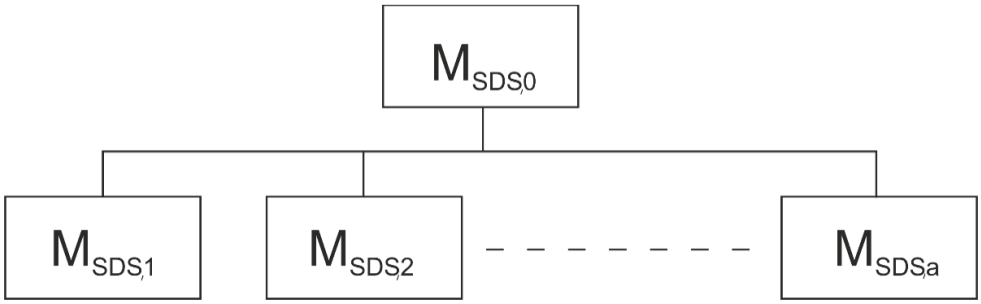

Рис. 2.1. Архітектура самоорганізованої розподіленої системи

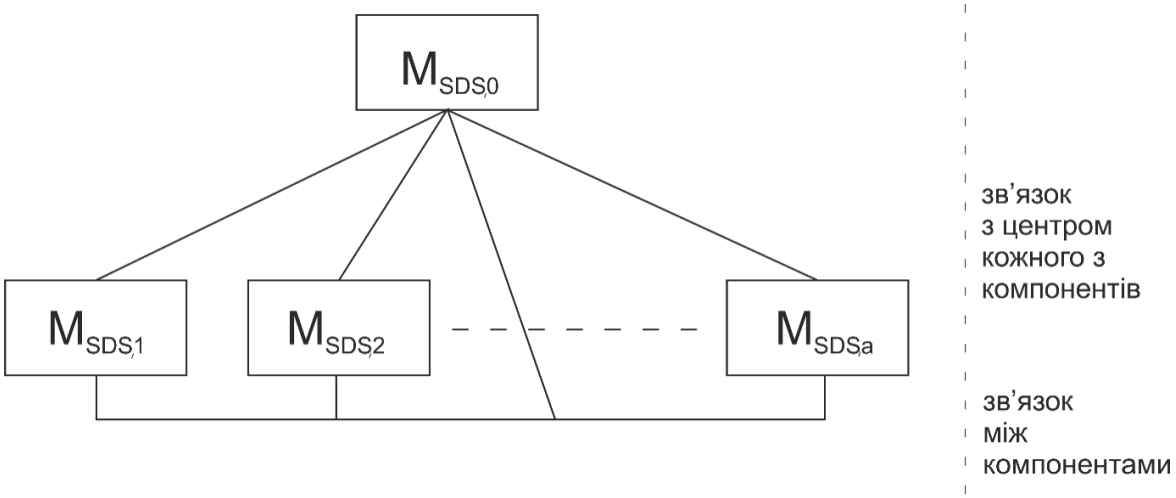

Рис. 2.2. Архітектура самоорганізованої розподіленої системи з відображенням центру системи в якості компоненти

В представленій архітектурі самоорганізованої розподіленої системи, на відміну від відомих рішень, удосконалено внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час. Це дало

змогу розподілити частину задач з центру на нижчий рівень ієрархії для прийняття рішення в залежності від застосовуваних методів з виявлення аномалій в конкретній комп'ютерній станції. Крім того, поділ задач центру на частини в залежності від їх призначення і за умови відсутності компоненти з вищим рівнем ієрархії, в якій знаходиться частину центру всієї самоорганізованої розподіленої системи, дає змогу здійснювати обмін повідомленнями про виявленні джерела об'єктів, які провокують аномальні прояви. Це є важливим в контексті специфіки вирішуваних задач системою та умов її функціонування, зокрема при руйнуючих впливах зловмисного програмного забезпечення. Зображення місць розміщення центру прийняття рішень в розроблюваній архітектурі самоорганізованої розподіленої системи представлено на рис. 2.3.

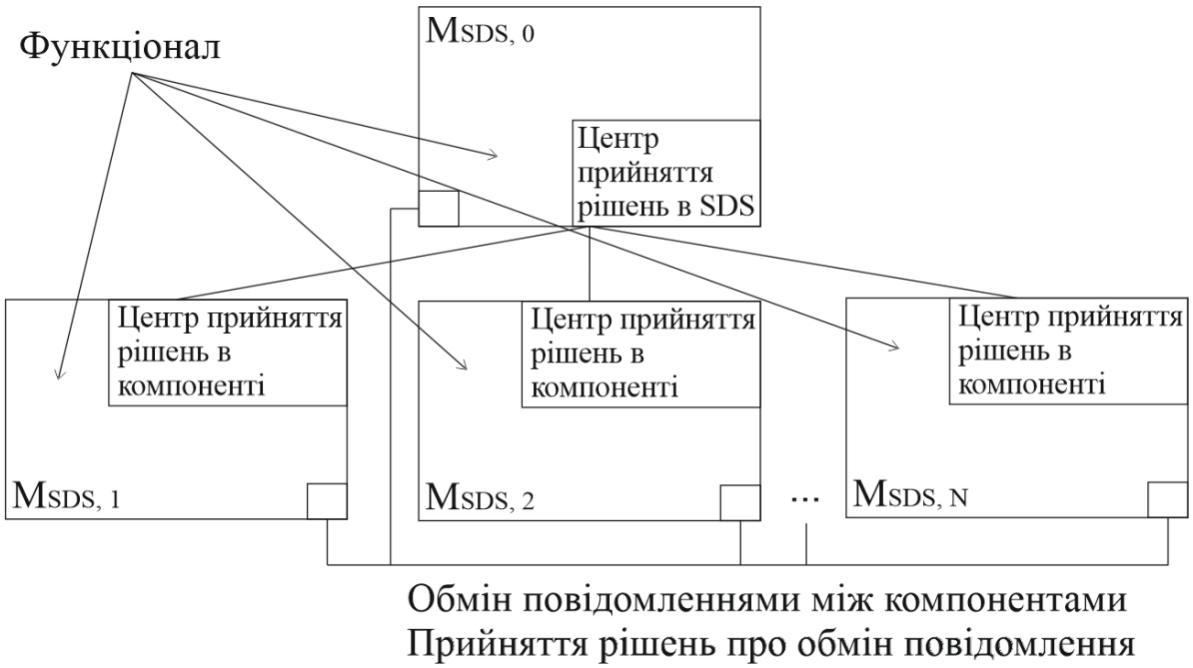

Рис. 2.3. Місця центру прийняття рішень в архітектурі самоорганізованої розподіленої системи

В результаті спроєктована архітектура самоорганізованої розподіленої системи дає змогу нарощувати її можливості за рахунок наповнення імплементованими методами з виявлення аномалій в комп'ютерних системах. В архітектурі системи розподілені функції її центру між компонентами, що пришвидшує обробку подій за рахунок здійснення обробки безпосередньо у вузлі мережі, в якому відбувся аналіз

аномального прояву. Крім того, система спроєктована таким чином, що її компоненти можуть обмінюватись результатами обробки аномальних проявів та виявленими їх джерелами.

2.2. Метод підтримки цілісності самоорганізованої розподіленої системи

Розподілення компонентів самоорганізованої розподіленої системи актуалізує проблему забезпечення цілісності системи та ефективної взаємодії компонентів, оскільки всі компоненти розміщені у різних вузлах в мережі. Крім того, підтримка цілісності самоорганізованої розподіленої системи за рахунок організації ефективної комунікації між ними є важливим завданням і не тільки за нормальних умов. Особливої ваги ефективність в організації такої комунікації в підтримці цілісності системи набуває за умов руйнуючих впливів зловмисного програмног забезпечення та комп'ютерних атак

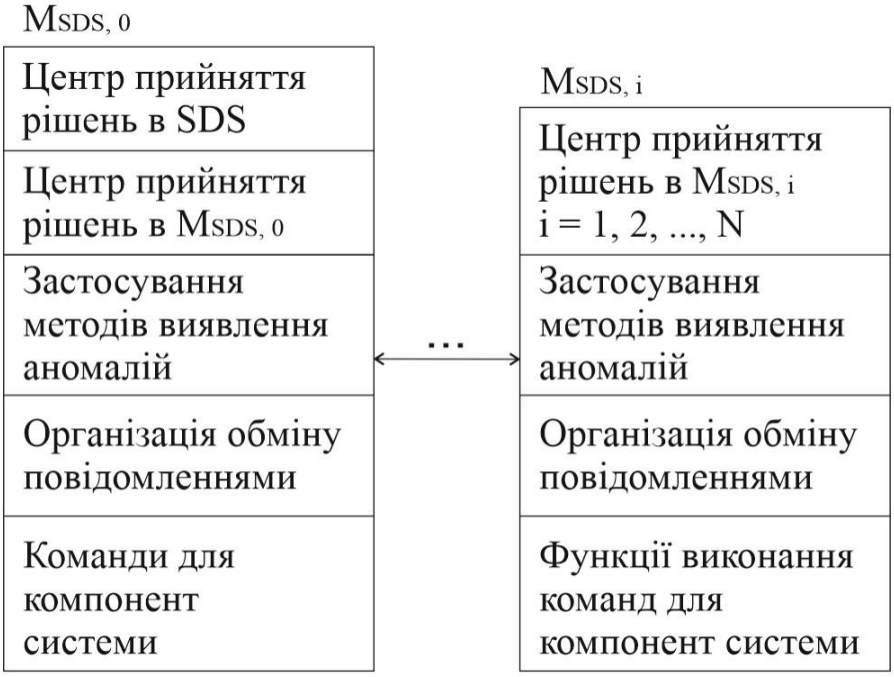

Рис. 2.4. Архітектура компонент системи та зв'язків між ними

Тому, метою підтримк цілісності самоорганізованої розподіленої системи за рахунок ефективної комунікації між компонентами повинен базуватись на невідомому для зловмисників мережному протоколі і мати набір варіантів подальших кроків всієї системи за настання відповідних

умов. Інтеграція цих двох складових в методі підтримки цілісності необхідна для покращення безпеки безпосередньо для самої системи порівняно з іншими функціонуючими системами функціонуючими за відомими мережними протоколами.

Розглянемо спочатку формування мережного протоколу для організації взаємодії компонентів. Врахуємо архітектурні особливості системи в тій частині, що відноситься до розподілення центрів прийняття рішень. Фактично саме з центрів прийняття рішень будуть надходити вказівки про пересилання даних. Тому, враховуючи три типи зв'язків між компонентами системи, які залежать від рівнів ієрархії на яких вони розміщені, отримуємо різні пари елементів по три компоненти в кожному так:

1) $(c_{SDS,0}, c_{SDS,i}, 1)$ – відображає передачу команди з центру верхнього рівня, який приймаємо таким позначенням як $c_{SDS,0}$, $c_{SDS,0} \in M_{DS,0}$, до центрів системи в компонентах, які знаходяться на нижньому рівні, тобто до центрів з позначенням $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$, де $i = 1, 2, \dots, N$;

2) $(c_{SDS,i}, c_{SDS,0}, 2)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру верхнього рівня $c_{SDS,0}$ для здійснення їх обробки;

3) $(c_{SDS,i}, c_{SDS,0}, 3)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті до центру верхнього рівня $c_{SDS,0}$;

4) $(c_{SDS,i}, c_{SDS,j}, 4)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$;

5) $(c_{SDS,i}, c_{SDS,j}, 5)$ – відображає передачу повідомлення з центру

нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i, i = 1, 2, \dots, N, j = 1, 2, \dots, N$;

6) $(c_{SDS,i}, c_{SDS,j}, 6)$ – відображає передачу повідомлення (при наявності відомостей про відсутність компоненти в системі з центром верхнього рівня, тобто фактичного центру прийняття рішень в системі) з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i, i = 1, 2, \dots, N, j = 1, 2, \dots, N$;

7) $(c_{SDS,i}, c_{SDS,j}, 7)$ – відображає передачу повідомлення (при наявності відомостей про відсутність компоненти в системі з центром верхнього рівня, тобто фактичного центру прийняття рішень в системі) з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{DS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i, i = 1, 2, \dots, N, j = 1, 2, \dots, N$;

8) $(c_{SDS,0}, c_{SDS,j}, 8)$ – відображає передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$, $c_{SDS,0} \in M_{DS,0}$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру нижнього рівня $c_{SDS,j}$ для здійснення їх обробки при цьому $j = 1, 2, \dots, N$;

9) $(c_{SDS,0}, c_{SDS,j}, 9)$ – відображає передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$, $c_{SDS,0} \in M_{DS,0}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру нижнього рівня $c_{SDS,j}$ для здійснення їх обробки при цьому $j = 1, 2, \dots, N$;

10) $(c_{SDS,i}, c_{SDS,0}, 10)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ до центру верхнього

рівня $c_{SDS,0}$, $c_{SDS,0} \in_{SDS,0}$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи;

Всі повідомлення між компонентами системи обов'язково обробляються центрами прийняття рішень в компонентах, незалежно від того до якого рівня ієрархії вони відносяться, і тільки після обробки повідомлень чи схвалення отриманих команд ними вони піддаються обробці чи виконанню іншими частинами компонент системи.

Команди, які надходять з центру верхнього рівня системи до центру системи нижнього рівня стосуються наступних подій чи станів: призупинити роботу компоненти; обробити дані характерних ознак для дослідження на аномалії; відновити роботу компоненти; надіслати компоненті з центром нижнього рівня повідомлення або вказану команду, тобто здійснити непряме звернення до іншої компоненти системи; надати данні про поточний стан комп'ютерної станції, тобто про профіль її характерних ознак та досліджуваних ознак на аномалії. Крім команд, з центру верхнього рівня системи можуть надходити повідомлення з інформацією, яку треба опрацювати, переслати в іншу компоненту чи зберігати.

Граф з дугами переходів, в якому враховуються три типи зв'язків між компонентами системи, що залежать від рівнів ієрархії на яких вони розміщені, зображено на рис. 2.5.

Відображення зв'язків між компонентами розподіленої системи та подіями, які можуть бути опрацьовані системою і задані елементами (1-16), на графі з дугами переходів є завершеним і не містить висячих вершин першого ступеня, тому описані події заданими елементами є достатніми для забезпечення функціонування розподіленої системи та можуть бути імплементовані в кроках методу підтримки цілісності самоорганізованої розподіленої системи. Додавання нових події, які можуть відбуватись в системі чи оброблятись в ній, є можливим, бо зв'язки в графі відображають замкнутість всіх подій, а їх збільшення фактично дозволятиме нарощувати функціональні можливості самої системи, оскільки кількість

компонентів при цьому не збільшуватиметься.

Перелік подальших кроків системи визначатиметься станами, в які може входити самоорганізована розподілена система або її компоненти. Також, ці стани залежатимуть від кількості активних компонентів системи, стану в комп'ютерних станціях в мережі. І, при цьому, ці стани будуть в часовому вимірі проміжними, оскільки система динамічно змінюватиме свою архітектуру та переходитиме зі стану в стан. Стани залежать від станів компонентів системи, причому як активних, так і вимкнених або вилучених системою.

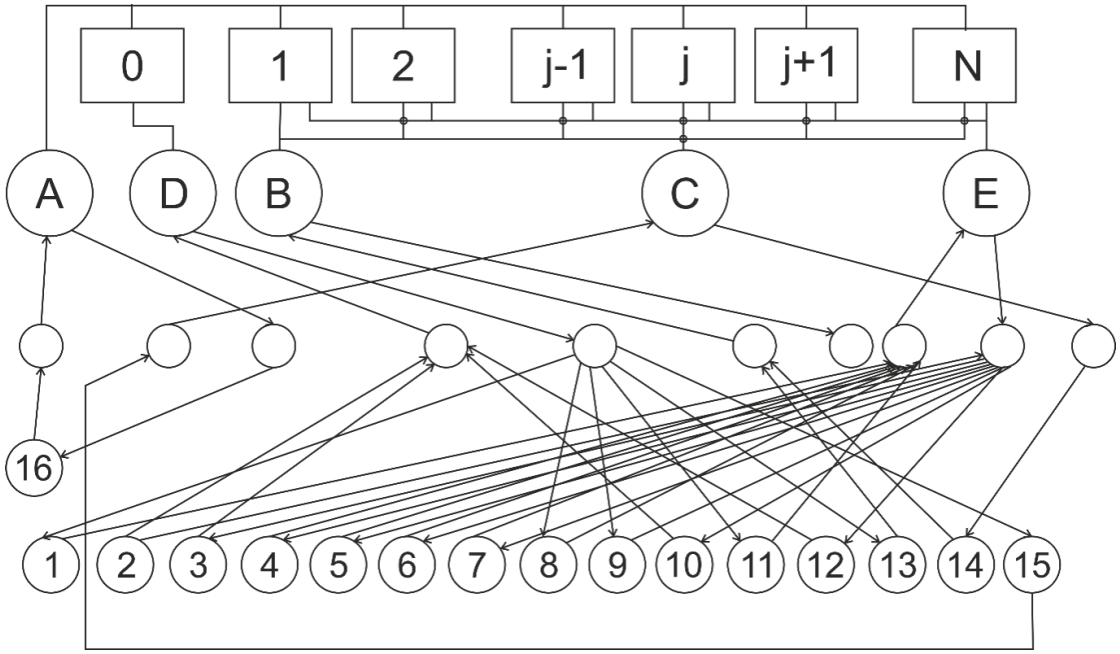

Рис. 2.5. Граф з дугами переходів

Метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах включає такі ітераційні кроки:

- крок 1: виконання $(c_{SDS,i}, c_{SDS,0}, 10)$ передачі повідомлення з центру нижнього рівня до центру верхнього рівня, $c_{SDS,0} \in M_{DS,0}$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи;

- крок 2: виконання $(c_{SDS,0}, c_{SDS,i}, 1)$ для передачі команди з центру верхнього рівня до центрів системи в компонентах, які знаходяться на нижньому рівні, і отримання підтвердження про отримання команди;

- крок 3: виконання команди в компоненті з центром нижчого рівня і надсилання звіту компоненті системи з центром вищого рівня;

- крок 4: виконання $(c_{SDS,i}, c_{SDS,0}, 2)$ та виконання $(c_{SDS,i}, c_{SDS,j}, 4)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів верхнього та нижнього рівнів для здійснення їх обробки;

- крок 5: виконання $(c_{SDS,i}, c_{SDS,0}, 3)$ та виконання $(c_{SDS,i}, c_{SDS,j}, 5)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про результати обробки аномалії до центрів верхнього та нижнього рівнів для здійснення їх обробки;

- крок 6: виконання $(c_{SDS,i}, c_{SDS,j}, 6)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів нижнього рівнів для здійснення їх обробки за умови відсутності центру верхнього рівня;

- крок 7: виконання $(c_{SDS,i}, c_{SDS,j}, 7)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про результати обробки аномалії до центрів нижнього рівнів для здійснення їх обробки за умови відсутності центру верхнього рівня;

- крок 8: виконання $(c_{SDS,0}, c_{SDS,j}, 8)$ для надсилання повідомлення з центру верхнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів нижнього рівнів для здійснення їх обробки;

- крок 9: виконання $(c_{SDS,0}, c_{SDS,j}, 9)$ для надсилання повідомлення з центру верхнього рівня, в якому міститься інформація про результати обробки аномалії до центрів нижнього рівнів для здійснення їх обробки.

Схема методу підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах полягає в тому, що в його кроках враховано стани компонентів системи, переходи між компонентами і завдяки закладеним в нього крокам вся розподілена система

може визначати свої подальші кроки. Основні кроки (1-15) методу виконуються не послідовно, а відповідають ітераційній схемі з одночасним паралельним виконанням певних кроків в різних компонентах одночасно. Тобто певна кількість кроків може виконуватись паралельно. Частина кроків може в певний момент часу не виконуватись. Перехід системи до визначених подальших станів відбувається на основі виконання певних кроків методу і залежить від сукупності можливих заданих станів, але фактично система і її компоненти переходять до виконання визначених кроків методу, тобто переходи до наступних станів системи і компонентів відповідають визначеним центром системи для виконання наступних кроків методу. Це надає змогу реалізувати за рахунок такого підходу таку характеристику системи як самоорганізованість, що на відміну від інших методів, які враховують для переходів дискретні стани системи або інтервальні проміжки для визначення станів системи чи її компонентів, враховує для переходів в якості мети для наступних дій виконання кроків методу. Завершенням виконання кроків методу за умов коректного вимкнення комп'ютерних станцій та програмного забезпечення компонентів системи буде завершення функціонування самоорганізованої розподіленої системи.

Таким чином, розроблено метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах, що враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи. Це дозволило будувати системи, які є централізованими та самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЙОГО ЗАСТОСУВАННЯ

3.1 Самоорганізована розподілена система виявлення аномалії в комп’ютерних системах

Самоорганізовану розподілену систему виявлення аномалії в комп’ютерних системах згідно розробленої її архітектури, методу підтримки цілісності та методу виявлення аномалій реалізуємо проміжним програмним забезпечення, що об’єнуватиме в одне ціле комп’ютерні станції в мережі, з двома типами інтерфейсу: для компоненти, в якій міститься центр прийняття рішень верхнього рівня ієрархії: для компонент, в яких міститься центр прийняття рішень нижнього рівня ієрархії. Зображення віконної форми інтерфейсу компоненти розподіленої системи, в якій міститься центр верхнього рівня ієрархії, представлено на рис. 3.1.

Таким чином, розроблене програмне забезпечення проміжного рівня підтверджує можливість реалізації запропонованих рішень.

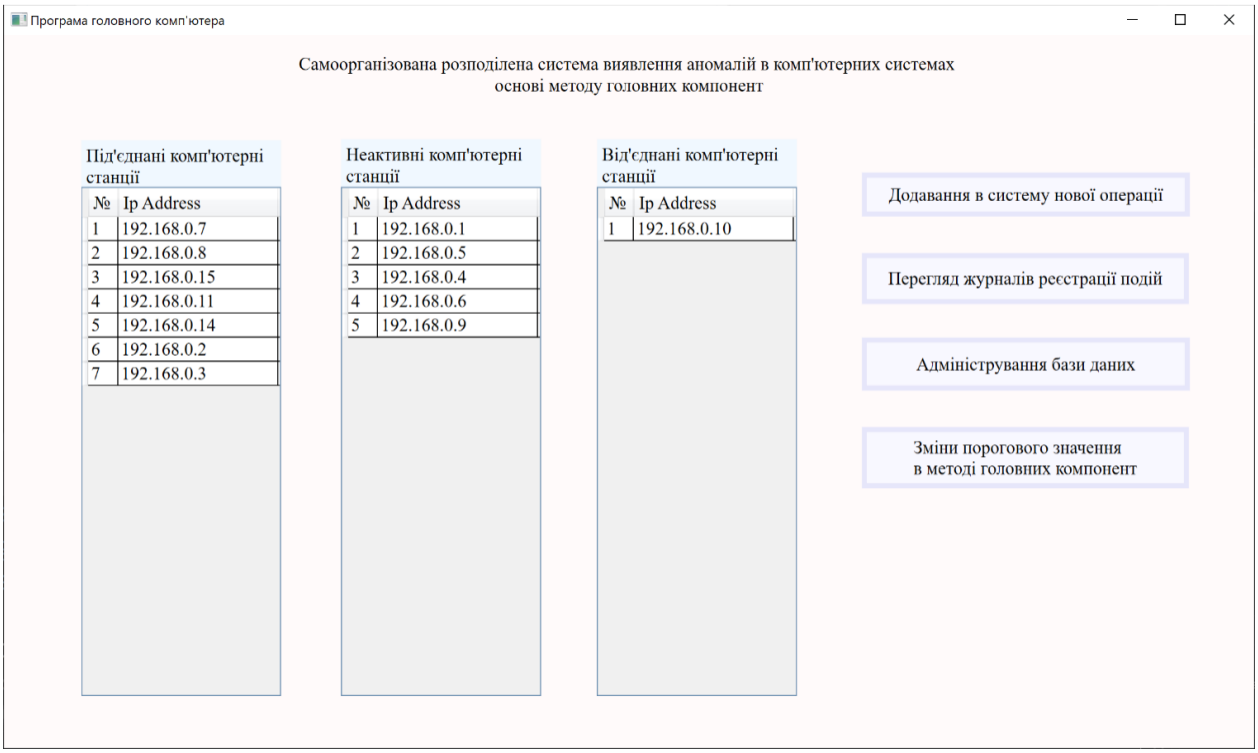

Рис. 3.1. Віконна форма інтерфейсу розробленої розподіленої системи

3.2. Експериментальні дослідження з використання самоорганізованої розподіленої системи

Метою експериментальних досліджень є дослідження ефективності функціонування самоорганізованої розподіленої системи та достовірність виявлення аномалії в комп'ютерних системах. Дослідження ефективності функціонування самоорганізованої розподіленої системи є необхідним для встановлення виконання нею функцій з підтримки цілісності та координації роботи компонентів системи. Достовірність виявлення аномалії в комп'ютерних системах потребує дослідження для встановлення можливості її використання в реальних умовах. Частина розподіленої системи, яка відповідає за виявлення аномалії, імплементована, як відповідний метод в неї, і її тестування дозволить оцінити достовірність з виявлення аномалії в комп'ютерних системах.

Тому, розглянемо тестування розподіленої системи спочатку з відімкненим модулем, що відповідає за виявлення аномалії. Показники, які досліджуватимемо згрупуємо за такими характеристиками: час комунікації між окремими компонентами; час комунікації між компонентами системи і компонентою, в якій розміщено центр прийняття рішень вищого рівня ієрархії; час комунікації між компонентами в залежності від кількості активних компонентів, які формують систему; час, який витрачено на роботу, коли в системі здійснено розподілення центру, тобто тестування з розподіленим центром, і час, який витрачений, коли центр не розподіляється між рівнями ієрархії, а знаходиться повністю в одній компоненті.

Експериментальні дослідження з розробленою самоорганізованою розподіленою системою проведено в локальній комп'ютерній мережі, створеній за технологією Ethernet з швидкістю передачі даних 1 Гб/с між вузлами в мережі. Оскільки повідомлення для передачі між компонентами системи містять дуже невелику кількість інформації, то вони формуються в короткі пакети і вважатимемо, що кожне з них передавалось одним пакетом обсягом 64 байти.

Досліджувана реалізована самоорганізована розподілена система містила вісім компонентів, в одній з яких знаходився центр прийняття рішень вищого рівня ієрархії. Експериментальні дослідження проводились протягом 50 діб окремо для випадку, коли центр прийняття рішень був розподілений між всіма компонентами з врахуванням двох рівнів ієрархії і так само 50 діб для випадку, коли центр прийняття рішень був розміщений тільки в одній компоненті. На протязі всього часу експерименту було здійснено фіксування часу відправки повідомлень, їх номеру та фіксування часу отримання. Це здійснювалось для того, щоб провести дослідження витрат часу на комунікацію в середині самої системи. Результати проведених експериментальних досліджень підтверджують збільшення кількості переданих повідомлень для випадку, коли час витрачено на роботу системи з розподіленням центру, тобто тестування з розподіленим центром, порівняно для випадку, коли час витрачено на роботу системи без розподілення центру між компонентами. Крім того, час обробки переданих пакетів з повідомленнями зростає, бо для випадку, коли час витрачено на роботу системи з розподіленням центру, повідомлення від компонент системи паралельно в один час надходять до компоненти, в якій розміщено центр прийняття рішень і розсилаються всім компонентам, на відміну від випадку, коли центр прийняття рішень, який представлений тільки в одній компоненті самостійно приймає рішення про комунікацію з рештою компонент, що зменшує суттєво кількість повідомлень між компонентами. Але при цьому часові витрати на передачу повідомлень є суттєво невеликими в обох випадках, тому використання архітектури з розподіленням центру між рівнями ієрархії в двох різних типах компонент системи є єфективним, що підтверджено результатами експериментів.

Отримані коефіцієнти підтверджують ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп’ютерній мережі. У другому випадку не великі значення коефіцієнтів обгрунтовуються тим, що тривалий час обслуговувались лише окремі компоненти системи, а більшість компонент систем функціонувала. Це доводить ефективність використання

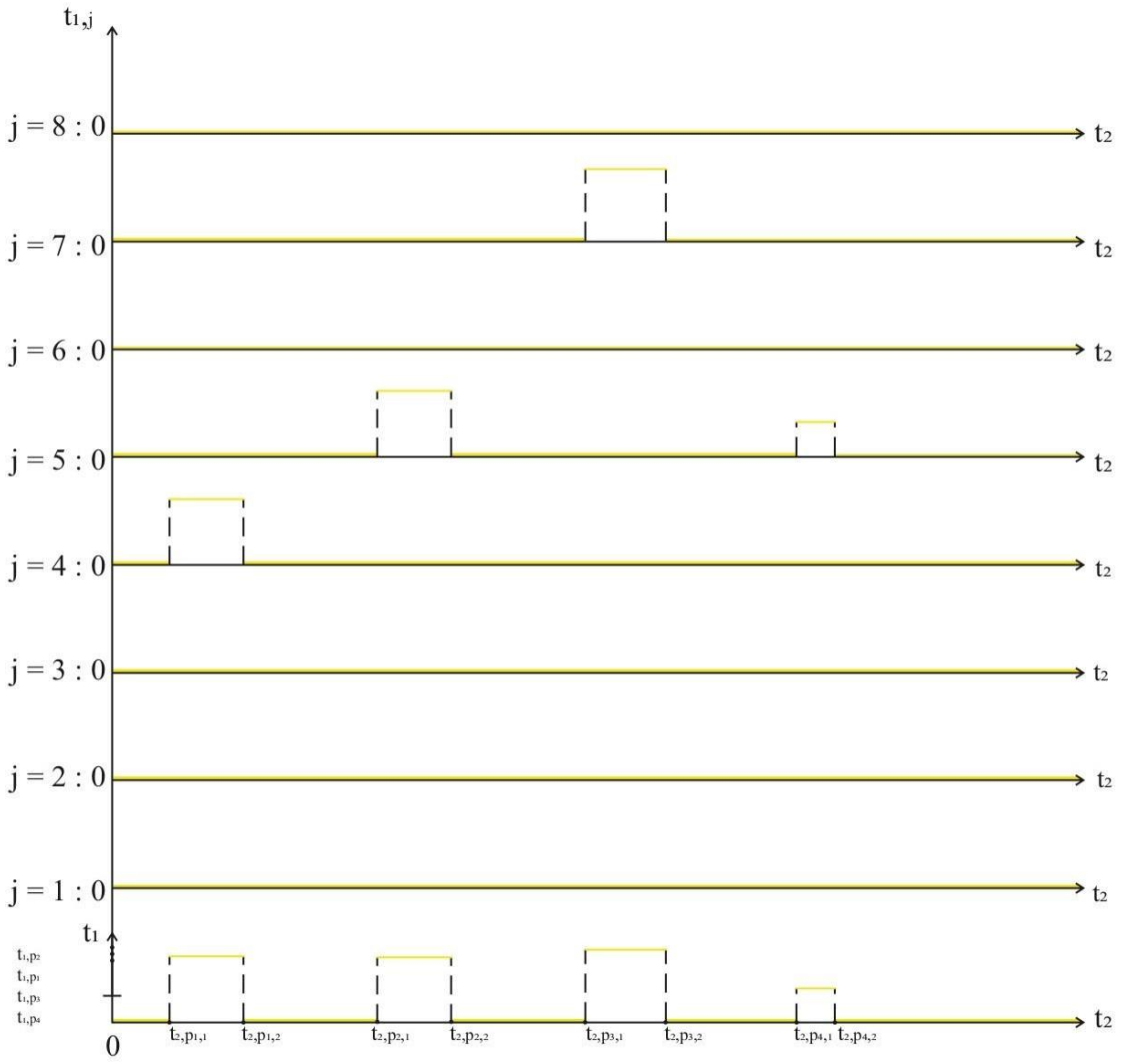

Рис. 3.2. Часові діаграми за результатами проведених експериментальних досліджень

Таблиця 3.1. Значення коефіцієнтів K_1 і K_2

№ з./п.	Час обслуговування компонентів системи користувачем або системним адміністратором був суттєво меншим часу роботи всієї системи	Час обслуговування компонентів системи користувачем або системним адміністратором був суттєво більшим часу роботи всієї системи
	Випадок 1	Випадок 2
K_1	0,00987567	0,04873418
K_2	0,00986972	0,04873326

Для оцінки ефективності запропонованих рішень розроблено методику розрахунку ефективності використання самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах. В ній використано два введені критерії, результати застосування яких визначаються через відповідні коефіцієнти. Крім того, отримані значення коефіцієнтів, які розраховані за певний проміжок часу враховуються при визначенні подальших дій системи.

Розроблене програмне забезпечення для забезпечення функціонування самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах підтверджує можливість реалізації запропонованих рішень.

Проведені експериментальні дослідження з розробленою реалізацією самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах згідно отриманих коефіцієнтів підтверджують ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп’ютерній мережі.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено самоорганізовану розподілену систему виявлення аномалій в комп'ютерних системах згідно методу головних компонент для покращення ефективності виявлення зловмисного програмного забезпечення та комп'ютерних атак.

При цьому отримано такі основні результати:

1. Встановлено, що виявлення зловмисного програмного забезпечення та комп'ютерних атак у локальних комп'ютерних мережах згідно досліджених методів та засобів виявлення може бути реалізовано методами виявлення аномалії в комп'ютерних системах та створенням розподілених систем виявлення аномалії.

2. Удосконалено архітектуру самоорганізованої розподіленої системи, в якій на відміну від відомих рішень, удосконалено внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії. Результатом так спроєктованої архітектури самоорганізованої розподіленої системи є можливість нарощувати її функціонал за рахунок наповнення імплементованими методами з виявлення аномалій в комп'ютерних системах. Система спроєктована таким чином, що її компоненти можуть обмінюватись результатами обробки аномалій та виявленими їх джерелами.

3. Розроблений метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stetsyuk M., Bedratyuk L., Savenko B., Stetsyuk V., Savenko O. Providing the Resilience and Survivability of Specialized Information Technology Across Corporate Computer Networks. CEUR-WS. 2020. Vol. 2623. P. 219-238.
2. Nicheporuk A., Paiuk V., Savenko B., Savenko O., Geidarova O. Detecting Software Malicious Implant Based on Anomalies Research on Local Area Networks. CEUR-WS. 2020. Vol. 2623. P. 194-207.
3. Лукова-Чуйко Н. В. Методологічні основи забезпечення функціональної стійкості розподілених інформаційних систем до кібернетичних загроз: автореф. дис. ... д-ра техн. наук: 05.13.06, Київ, 2018, 40 с.
4. Mohiuddin Ahmed, Abdun Naser Mahmood, Jiankun Hu. A survey of network anomaly detection techniques. Journal of Network and Computer Applications Vol. 60, January 2016. P. 19-31.
5. Xiang Yu, Hui Lu, Xianfei Yang, Ying Chen, Haifeng Song, Jianhua Li, Wei Shi An adaptive method based on contextual anomaly detection in Internet of Things through wireless sensor networks. International Journal of Distributed Sensor Networks 2020. Vol. 16(5) DOI: 10.1177/1550147720920478
6. Goldstein M., Uchida S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. 2016. PLOS ONE 11(4): e0152173. https://doi.org/10.1371/journal.pone.0152173
7. Hayes, M.A., Capretz, M.A. Contextual anomaly detection framework for big sensor data. Journal of Big Data 2**,** 2. 2015. https://doi.org/10.1186/s40537-014-0011-y
8. Liu, L., Hu, M.; Kang, C., Li, X. Unsupervised Anomaly Detection for Network *Data Streams in Industrial Control Systems.* Information *2020,* 11, *105*. https://doi.org/10.3390/info11020105
9. Solarz A., Bilicki M., Gromadzki M., Pollo A. , Durkalec A., Wypych M. Automated noveltydetection in the WISE surveywithone-classsupport vector

machines. Published online: 05 October 2017. DOI: 10.1051/0004-6361/201730968

10. Mukrimah Nawir, Amiza Amir, Naimah Yaakob, Ong Bi Lynn. Effective and efficient network anomaly detection system using machine learning algorithm. Bulletin of Electrical Engineering and InformaticsVol.8, No.1, March 2019, pp. 46~51ISSN: 2302-9285, DOI: 10.11591/eei.v8i1.1387
11. Anta A., Hadjistasi T., Nicolaou N. et al. Tractable low-delay atomic memory. Distrib. Comput. 34, 33–58 (2021). https://doi.org/10.1007/s00446-020-00379-y
12. Ouyang L., Huang Y., Wei H., Lu J. Achieving Probabilistic Atomicity With Well-Bounded Staleness and Low Read Latency in Distributed Datastores. in IEEE Transactions on Parallel and Distributed Systems, vol. 32, № 4, pp. 815-829, 1 April 2021, doi: 10.1109/TPDS.2020.3034328.
13. Lakshman A. and Malik P. Cassandra: A decentralized structured storage system, *SIGOPS Operating Syst. Rev.*, vol. 44, no. 2, pp. 35-40, Apr. 2010.
14. Ganesh, C., Patra, A. Optimal extension protocols for byzantine broadcast and agreement. Distrib. Comput. 34, 59–77 (2021). https://doi.org/10.1007/s00446-020-00384-1
15. Huang, Z., Radunovic, B., Vojnovic, M. et al. Communication complexity of approximate maximum matching in the message-passing model. Distrib. Comput. 33, 515–531 (2020). https://doi.org/10.1007/s00446-020-00371-6
16. Czumaj, A., Konrad, C. Detecting cliques in CONGEST networks. Distrib. Comput. 33, 533–543 (2020). https://doi.org/10.1007/s00446-019-00368-w
17. Min B., Varadharajan V. (2014) Feature-Distributed Malware Attack: Risk and Defence. In: Kutyłowski M., Vaidya J. (eds) Computer Security - ESORICS 2014. ESORICS 2014. Lecture Notes in Computer Science, vol 8713. Springer, Cham. https://doi.org/10.1007/978-3-319-11212-1_26
18. Nath H. V., Mehtre B.M. (2014) Static Malware Analysis Using Machine Learning Methods. In: Martínez Pérez G., Thampi S.M., Ko R., Shu L. (eds) Recent Trends in Computer Networks and Distributed Systems Security. SNDS 2014. Communications in Computer and Information Science, vol 420. Springer, Berlin, Heidelberg.

19. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 97–120. Springer, Berlin (2009)
20. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations, and Advanced Topics, vol. 19. Wiley, New York (2004)
21. Chen, H.-L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Nat. Comput. **13**, 517–534 (2014)
22. Censor-Hillel, K., Parter, M. & Schwartzman, G. Derandomizing local distributed algorithms under bandwidth restrictions. *Distrib. Comput.* **33,** 349–366 (2020). https://doi.org/10.1007/s00446-020-00376-1
23. Barenboim, L.: Deterministic (Δ+1)-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In: PODC, pp. 345–354 (2015)
24. Barenboim, L., Elkin, M., Gavoille, C.: A fast network-decomposition algorithm and its applications to constant-time distributed computation. In: SIROCCO, pp. 209–223 (2015)
25. Benjamini, I., Gurel-Gurevich, O., Peled, R.: On k-wise independent distributions and Boolean functions. arXiv preprint arXiv:1201.3261 (2012)
26. Antoniadis, K., Blanchard, P., Guerraoui, R. *et al.* The entropy of a distributed computation random number generation from memory interleaving. *Distrib. Comput.* 31, 389–417 (2018). https://doi.org/10.1007/s00446-017-0311-5
27. Mohiuddin Ahmed, Abdun Naser Mahmood, Jiankun Hu. A survey of network anomaly detection techniques. Journal of Network and Computer Applications Vol. 60, January 2016. P. 19-31.
28. Xiang Yu, Hui Lu, Xianfei Yang, Ying Chen, Haifeng Song, Jianhua Li, Wei Shi An adaptive method based on contextual anomaly detection in Internet of Things through wireless sensor networks. International Journal of Distributed Sensor Networks 2020. Vol. 16(5) DOI: 10.1177/1550147720920478
29. Goldstein M., Uchida S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. 2016. PLOS ONE 11(4): e0152173. https://doi.org/10.1371/journal.pone.0152173

30. Hayes, M.A., Capretz, M.A. Contextual anomaly detection framework for big sensor data. Journal of Big Data 2**,** 2. 2015. https://doi.org/10.1186/s40537-014-0011-y
31. Liu, L., Hu, M.; Kang, C., Li, X. Unsupervised Anomaly Detection for Network *Data Streams in Industrial Control Systems.* Information *2020*, 11, *105*. https://doi.org/10.3390/info11020105
32. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 97–120. Springer, Berlin (2009)
33. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations, and Advanced Topics, vol. 19. Wiley, New York (2004)
34. Chen, H.-L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Nat. Comput. **13**, 517–534 (2014)
35. Censor-Hillel, K., Parter, M. & Schwartzman, G. Derandomizing local distributed algorithms under bandwidth restrictions. *Distrib. Comput.* **33,** 349–366 (2020). https://doi.org/10.1007/s00446-020-00376-1
36. Barenboim, L.: Deterministic (Δ+1)-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In: PODC, pp. 345–354 (2015)

Додаток А

Лістинг програми

ЛІСТИНГ
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ САМООРГАНІЗОВАНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ

```
using DistributedSystem.LowerCenter.Database;
using
DistributedSystem.LowerCenter.Database.Entities;
using
DistributedSystem.LowerCenter.Database.Repository;
using Microsoft.EntityFrameworkCore;
using
System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace
DistributedSystem.LowerCenter.Anomaly {
   public class AnomalyService :
   IAnomalyService {
      private readonly IRepository<AnomalyEntity> _repository;

      public
      AnomalyService()
      {
         _repository = new
      Repository<AnomalyEntity>(); }

      public async Task<List<long>>
      GetDetectedAnomaliesIdList() {
         return await _repository.Entities.Where(e => e.Status ==
                                 AnomalyStatus.DETECTED) .Select(e => e.Id)
                                 .ToListAs
      ync(); }

      public async Task<List<long>>
      GetDetectedPotentialAnomaliesIdList() {
         return      await      _repository.Entities.Where(e      =>      e.Status
                     == AnomalyStatus.POTENTIAL)
                                 .Select(e =>
                                 e.Id)
                                 .ToListAsy
                                 nc();
      }
   }
}
using
System.Collections.Generic;
using
System.Threading.Tasks;

namespace
```

```
DistributedSystem.LowerCenter.Anomaly {
   public interface
   IAnomalyService {
      Task<List<long>> GetDetectedAnomaliesIdList();
      Task<List<long>>
      GetDetectedPotentialAnomaliesIdList();


}

using
DistributedSystem.LowerCenter.Database.Entities;
using
DistributedSystem.LowerCenter.Database.Repository;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace
DistributedSystem.LowerCenter.Command {
   public class CommandService :
   ICommandService {
      private readonly IRepository<CommandEntity> _repository;

      public
      CommandService()
      {
         _repository = new
      Repository<CommandEntity>(); }

      public async Task<int> GetCommandId(string
      commandName) {
         return await _repository.Entities.Where(e => e.Name ==
commandName).Select(e => e.Id).FirstOrDefaultAsync();
      }
   }
}

using System.Threading.Tasks;

namespace
DistributedSystem.LowerCenter.Command {
   public interface
   ICommandService {
      Task<int> GetCommandId(string
   commandName); }
}

using System;

namespace
DistributedSystem.LowerCenter.Database.Entities {
   public class
   AnomalyEntity {
      public long Id { get; set;
      } public string Name {
```

```
            get; set; }
            public string Description { get; set;
            } public AnomalyStatus Status {
            get; set; } public DateTime
            DetectedOn { get; set; }
        }
    }
{
    public class
    CommandEntity {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get;
        set; }
    }
}

using System;

namespace
DistributedSystem.LowerCenter.Database.Entities {
    public class
    InvestigationResultEntity {
        public int Id { get; set; }
        public string Conclusion { get; set; }
        public DateTime CreatedOn { get;
        set; }

        public long AnomalyFK { get; set; }
        public AnomalyEntity Anomaly { get; set; }

        public int CommandFK { get; set; }
        public CommandEntity Command { get;
    set; } }
}

using System;

namespace
DistributedSystem.LowerCenter.Database.Entities {
    public class
    LogActivityEntity {
        public long Id { get; set; }
        public string Name { get;
        set; }
        public string Description { get; set; }
        public string MachineIP { get; set; }
        public DateTime CreatedOn { get;
        set; }
    }
}

namespace
DistributedSystem.LowerCenter.Database {
    public enum
    AnomalyStatus {
```

```
        DETECTED = 1,
        POTENTIAL = 2,
        INVESTIGATED =
        3, RESOLVED = 4
}
    using System;
    using
    System.Collections.Generic;
    using System.Linq;
    using
    System.Linq.Expressions;
    using
    System.Threading.Tasks;

    namespace
    DistributedSystem.LowerCenter.Database.Repository {
      public interface IRepository<T> where T :
      class {
        IQueryable<T> Entities {
        get; } T GetById(int id);
        IEnumerable<T> GetAll();
        IEnumerable<T> Find(Expression<Func<T, bool>>
        expression); Task Add(T entity);
        Task AddRange(IEnumerable<T>
        entities); Task Update(T entity);
        Task Remove(T entity);
        Task RemoveRange(IEnumerable<T>
      entities); }
    }

    using
    Microsoft.EntityFrameworkCore;
    using System;
    using
    System.Collections.Generic;
    using System.Linq;
    using
    System.Linq.Expressions;
    using
    System.Threading.Tasks;

    namespace
    DistributedSystem.LowerCenter.Database.Repository {
      public class Repository<T> : IRepository<T> where T :
      class {
        protected readonly AppDbContext _dbContext;

        public
        Repository() {
          _dbContext = new
        AppDbContext(); }

        public IQueryable<T> Entities => _dbContext.Set<T>();

        public virtual async Task Add(T
        entity) {
```

```
            await _dbContext.AddAsync(entity);
            await
            _dbContext.SaveChangesAsync();
        }

        public virtual async Task AddRange(IEnumerable<T>
        entities) {
await _dbContext.AddRangeAsync(entities);

        }

        public virtual async Task Update(T
        entity) {
            _dbContext.Update(entity);
            await
        _dbContext.SaveChangesAsync(); }

        public virtual async Task<IEnumerable<T>> Find(Expression<Func<T,
bool>> expression)
        {
            return await
        _dbContext.Set<T>().Where(expression).ToListAsync(); }

        public virtual async Task<IEnumerable<T>>
        GetAll() {
            return await
        _dbContext.Set<T>().ToListAsync(); }

        public virtual async Task<T>
        GetById(int id) {
            return await
        _dbContext.Set<T>().FindAsync(id); }

        public virtual async Task Remove(T
        entity) {
            _dbContext.Remove(entity);
            await
        _dbContext.SaveChangesAsync(); }

        public virtual async Task RemoveRange(IEnumerable<T>
        entities) {
            _dbContext.RemoveRange(entities
            ); await
            _dbContext.SaveChangesAsync();
        }
    }
}

using
DistributedSystem.LowerCenter.Database.Entities;
using Microsoft.EntityFrameworkCore;

namespace
DistributedSystem.LowerCenter.Database {
    public class AppDbContext :
    DbContext {
```

```
        public DbSet<AnomalyEntity> Anomalies { get;
        set; } public DbSet<CommandEntity> Commands
        { get; set; }
        public DbSet<InvestigationResultEntity> InvestigationResults { get;
        set; } public DbSet<LogActivityEntity> ActivityLogs { get; set; }

public AppDbContext(DbContextOptions<AppDbContext>
options)

        {
            Database.Migr
        ate(); }

        public
        AppDbContext()
        {

        }
    }
}

using
DistributedSystem.LowerCenter.Database.Entities;
using System.Threading.Tasks;

namespace
DistributedSystem.LowerCenter.InvestigationResult {
    public interface
    IInvestigationResultService {
        Task<InvestigationResultEntity> GetInvestigationResultByAnomalyId(long
    anomalyId); }
}

using
DistributedSystem.LowerCenter.Database.Entities;
using
DistributedSystem.LowerCenter.Database.Repository;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace
DistributedSystem.LowerCenter.InvestigationResult {
    public class InvestigationResultService :
    IInvestigationResultService {
        private readonly IRepository<InvestigationResultEntity> _repository;

        public
        InvestigationResultService()
        {
            _repository = new
        Repository<InvestigationResultEntity>(); }

        public async Task<InvestigationResultEntity>
GetInvestigationResultByAnomalyId(long anomalyId)
```

```
            {
                return await _repository.Entities.Where(e => e.Id
                == anomalyId).FirstOrDefaultAsync();
            }
        }
    }

    {
        public interface
        ILogActivityService {
            Task Log(LogActivityEntity
        newRecord); }
    }

    using
    DistributedSystem.LowerCenter.Database.Entities;
    using
    DistributedSystem.LowerCenter.Database.Repository;
    using System.Threading.Tasks;

    namespace
    DistributedSystem.LowerCenter.LogActivity {
        public class LogActivityService :
        ILogActivityService {
            private readonly IRepository<LogActivityEntity> _repository;

            public
            LogActivityService()
            {
                _repository = new
            Repository<LogActivityEntity>(); }

            public async Task Log(LogActivityEntity
            newRecord) {
                await
            _repository.Add(newRecord); }
        }
    }

    using System.Threading.Tasks;

    namespace
    DistributedSystem.LowerCenter.TcpClientSender {
        public interface
        ITcpClientSender {
            Task
            SendWorkReadinessMessageToMainCenter();
            Task
            SendDetectedPotentialAnomaliesToMainCenter();
            Task SendAnomalyProcessingResultToMainCenter(int
            anomalyId); Task SendShutDownMessageToMainCenter();
            Task
                        SendAnomalyProcessingResultToCurrentCenterIfMainCenterAbse
nt(int anomalyId);
            Task
        SendDetectedPotentialAnomaliesToCurrentCenter(); }
```

```
        }

        using
        DistributedSystem.LowerCenter.Anomaly;
        using
        DistributedSystem.LowerCenter.Command;
        using
        DistributedSystem.LowerCenter.Database.Entities;
        using
        DistributedSystem.LowerCenter.InvestigationResult;
        using DistributedSystem.LowerCenter.LogActivity;
using System;

        using
        System.Net.Sockets;
        using System.Text;
        using System.Threading.Tasks;

        namespace
        DistributedSystem.LowerCenter.TcpClientSender {
           public class TcpClientSender :
           ITcpClientSender {
              private readonly TcpClient _client;
              private readonly ILogActivityService
              _logActivityService; private readonly
              ICommandService _commandService; private
              readonly IAnomalyService _anomalyService;
              private readonly IInvestigationResultService _investigationResultService;

              private
              TcpClientSender() {
                 _client        =        new
                                TcpClient(Constants.MAIN_CENTER_IP_ADDRESS,
Constants.MAIN_CENTER_PORT);
                 _logActivityService = new
                 LogActivityService(); _commandService =
                 new CommandService(); _anomalyService
                 = new AnomalyService();
                 _investigationResultService = new
              InvestigationResultService(); }

              public async Task
              SendWorkReadinessMessageToMainCenter() {
                 t
                 r
                 y
                 {
                    NetworkStream stream = _client.GetStream();

                    while
                    (true)
                    {
                       var commandId = await _commandService.GetCommandId("Readiness");

                       byte[] data =
                       Encoding.Unicode.GetBytes(commandId.ToString());
```

```
                    stream.Write(data, 0, data.Length);
                }
            }
            catch
            (Exception ex)
            {
                string myHost = Dns.GetHostName();

                var exception = new
                LogActivityEntity() {
                    Name = "Exception",
                    MachineIP                                   =
                                                                (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                    Description =
                    ex.Message, CreatedOn
                    = DateTime.Now,
                };
        using
        System.Net.Sockets;
        using System.Text;
        using System.Threading.Tasks;

        namespace
        DistributedSystem.LowerCenter.TcpClientSender {
            public class TcpClientSender :
            ITcpClientSender {
                private readonly TcpClient _client;
                private readonly ILogActivityService
                _logActivityService; private readonly
                ICommandService _commandService; private
                readonly IAnomalyService _anomalyService;
                private readonly IInvestigationResultService _investigationResultService;

                private
                TcpClientSender() {
                    _client         =         new
                                    TcpClient(Constants.MAIN_CENTER_IP_ADDRESS,
Constants.MAIN_CENTER_PORT);
                    _logActivityService = new
                    LogActivityService(); _commandService =
                    new CommandService(); _anomalyService
                    = new AnomalyService();
                    _investigationResultService = new
                InvestigationResultService(); }

                public async Task
                SendWorkReadinessMessageToMainCenter() {
                    t
                    r
                    y
                    {
                        NetworkStream stream = _client.GetStream();

                        while
                        (true)
```

```
                {
                    var commandId = await _commandService.GetCommandId("Readiness");

                    byte[] data =
                    Encoding.Unicode.GetBytes(commandId.ToString());
                    stream.Write(data, 0, data.Length);
                }
            }
            catch
            (Exception ex)
            {
                string myHost = Dns.GetHostName();

                var exception = new
                LogActivityEntity() {
                    Name = "Exception",
                    MachineIP                                   =
                                                                (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                    Description =
                    ex.Message, CreatedOn
                    = DateTime.Now,
                };
        using
        System.Net.Sockets;
        using System.Text;
        using System.Threading.Tasks;

        namespace
        DistributedSystem.LowerCenter.TcpClientSender {
            public class TcpClientSender :
            ITcpClientSender {
                private readonly TcpClient _client;
                private readonly ILogActivityService
                _logActivityService; private readonly
                ICommandService _commandService; private
                readonly IAnomalyService _anomalyService;
                private readonly IInvestigationResultService _investigationResultService;

                private
                TcpClientSender() {
                    _client         =         new
                                    TcpClient(Constants.MAIN_CENTER_IP_ADDRESS,
Constants.MAIN_CENTER_PORT);
                    _logActivityService = new
                    LogActivityService(); _commandService =
                    new CommandService(); _anomalyService
                    = new AnomalyService();
                    _investigationResultService = new
                InvestigationResultService(); }

                public async Task
                SendWorkReadinessMessageToMainCenter() {
                    t
                    r
                    y
```

```
                {
                    NetworkStream stream = _client.GetStream();

                    while
                    (true)
                    {
                        var commandId = await _commandService.GetCommandId("Readiness");

                        byte[] data =
                        Encoding.Unicode.GetBytes(commandId.ToString());
                        stream.Write(data, 0, data.Length);
                    }
                }
                catch
                (Exception ex)
                {
                    string myHost = Dns.GetHostName();

                    var exception = new
                    LogActivityEntity() {
                        Name = "Exception",
                        MachineIP                                   =
                                                                    (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                        Description =
                        ex.Message, CreatedOn
                        = DateTime.Now,
                    };
```

Додаток Б

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Кваліфікаційна робота.doc	Пояснювальна записка. Документ Word.
Кваліфікаційна робота.pdf	Пояснювальна в форматі PDF
Программа	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація