МІНІСТЕРСТОВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри КІТ
_____ А. С. Савченко
«____» _____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТРА"

ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: «Метод діагностики розподіленої мережної системи»

Виконавець: студентка УС-211М Іванова Олена Анатоліївна
(студент, група, призвіще, ім'я, по батькові)

Керівник: доктор технічних наук, професор ВІНОГРАДОВ Микола Анатолійович
(науковій ступень, вчене звання, призвіще, ім'я, по батькові)

Нормоконтролер _____ Райчев І. Е.
(підпис) (П.І.Б.)

КИЇВ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій
Освітній ступінь **Магістр**
Напрям 6.050101 Комп'ютерні науки
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри

_____А. С. Савченко
« » _____2020 р.

ЗАВДАННЯ

на виконання дипломного проекту студентки

Іванової Олени Анатоліївни
(прізвище, ім'я, по батькові)

1.Тема проекту: «Метод діагностики розподіленої мережної системи» затверджена наказом ректора № 567/ст. від 16.03.2020р.

2.Термін виконання роботи: з 26.09.2019 р. по 08.02.2020 р.

3. Вихідні дані до роботи: розробка систми моніторингу розподіленної мережі

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, проект універсальної програмної бібліотеки на базі стеганографічних методів для забезпечення захисту інформаційних систем, висновок.

5. Перелік обов'язкового графічного матеріалу: огляд загальної структури стегосистем, структура формату PNG, структура формату JPEG, схема загального алгоритму роботи стегосистем, діаграма класів DLL бібліотеки «stegoDevLib» .

6. КАЛЕНДАРНИЙ ПЛАН

	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Аналіз літератури та джерел за темою дипломного проекту.		
2	Розробка та затвердження плану дипломного проекту.		
3	Проведення консультації з науковим керівником щодо створення першого розділу.		
4	Аналітичний огляд і постановка задачі. Порівняльний аналіз існуючих програмних реалізацій		
5	Розробка методу моніторингу		
6	Розробка системи моніторингу		
7	Висновки та оформлення пояснювальної записки дипломного проекту.		
8	Підписання необхідних документів у встановленому порядку.		
9	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту		

7. Дата видачі завдання: 06.05.2020 р.

Керівник дипломної роботи _____ Віноградов М.А. .
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Іванова О.А .
(підпис випускника) (П.І.Б.)

Реферат

Пояснювальна записка до дипломного проекту «Метод діагностики розподіленої мережної системи»: 65 с., 30 рис., 2 табл., 14 літературних джерел.

Ключові слова: УПРАВЛЯЮЧІ СИСТЕМИ, РОЗПОДІЛЕНІ МЕРЕЖІ, ПОШУК ВІДМОВ, МОНІТОРИНГ ТРАФІКУ, ІНФОРМАЦІЙНА БЕЗПЕКА, ДІАГНОСТИКА МЕРЕЖ .

Об’єкт дослідження: управління розподіленими мережними системами.

Предмет дослідження: технологія моніторингу розподілених мереж.

Мета роботи: розробка методу і алгоритмів моніторингу розподіленої мережної системи, заснованих на аналізі поведінки трафіку та фізичного стану хостів, та побудова системи моніторингу мережі на їх основі

Методи дослідження, технічні та програмні засоби: розробка програмних бібліотек, порівняльний аналіз, обробка літературних джерел.

Отримані результати та їх новизна:

Матеріали проекту можуть бути використані в розробках спеціалізованого програмного забезпечення або в освітніх закладах при висвітленні теми інформаційної безпеки управляючих систем.

.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ПЗ	Програмне забезпечення
ІС	Інформаційна система
ІБ	Інформаційна безпека
ФС	Файлова система
ОС	Операційна система

ВСТУП

З ростом рівня автоматизації, проникнення інформаційних технологій в усі сфери діяльності людини, спостерігається також і значне підвищення вимог до відмовостійкості та надійності інформаційних систем. А отже, дедалі більшого значення набувають питання розробки ефективних засобів моніторингу та діагностики інформаційних систем. У зв'язку з повсюдним використанням обчислювальних мереж і мереж передачі даних для організації взаємодії як між окремими робочими станціями (наприклад, при роботі в глобальній мережі Інтернет), так і взаємодії всередині корпоративних мереж та інших розподілених систем, заснованих на використанні обчислювальних мереж, гостро постають питання моніторингу стану подібних систем.

Зростання складності мереж ускладнює процес їх адміністрування, який включає в себе такі завдання, як:

- пошук та усунення несправностей апаратної частини;
- установка нових служб і протоколів;
- налаштування програмного забезпечення;
- пошук шляхів оптимізації роботи мережі.

Дуже важливим завданням при адмініструванні мереж є своєчасне реагування на виникаючі проблеми і їх усунення в мінімально можливі терміни. Для оперативного і ефективного вирішення завдань адміністрування необхідна достовірна інформація про роботу компонентів і систем мережі. Ця інформація може бути зібрана і оброблена, зокрема, за допомогою спеціальних програмних систем моніторингу. Вибір способів і об'єктів моніторингу мережі залежить від безлічі факторів - розміру IT-інфраструктури системи, конфігурації мережі, обладнання що використовується в ній, діючих сервісів і служб, конфігурації серверів і встановленого на них програмного забезпечення тощо.

Класичні системи моніторингу забезпечують безперервний моніторинг поточного стану вузлів, що входять до складу мережі, але в умовах жорстких вимог до їх відмовостійкості та надійності, до них пред'являються також вимоги щодо забезпечення можливостей прогнозування і діагностики стану обслуговуваних

інформаційних систем в короткостроковому і довгостроковому періодах. Також стає необхідною реалізація комплексного моніторингу, що включає аналіз як поточного стану вузлів, так і комплексний аналіз функціонування системи, а також прогнозування та діагностику потенційних проблем в системі.

Існує безліч методів і засобів для моніторингу мережевих підключень. Особливості їх використання залежать від цілей процесу, конфігурації мережі, файлової системи тощо

Основні спрямування моніторингу:

- Аналізатори протоколів. Використовуються виключно для контролю мережевого трафіку.
- Інтегровані системи управління та аналізу. Використовуються для моніторингу програмної та апаратної складових середовища. Забезпечують контроль певних програм, відрізків комунікацій і окремих пристроїв в мережі.
- Управління мережею. Так само як і інтегровані системи, збирає дані про мережеві процеси і про стан апаратного блоку, проте відстежують весь мережевий трафік.
- Кабельне обладнання. Використовуються при сертифікації і тестуванні кабельних мереж.

Метою роботи є розробка методу моніторингу розподіленої мережної системи, заснованих на аналізі поведінки трафіку та фізичного стану хостів, та побудова системи моніторингу мережі на їх основі.

Для досягнення поставленої мети були вирішені наступні завдання:

1. Дослідження і аналіз сучасних підходів до організації систем моніторингу мережі.
2. Розробка методу і алгоритмів моніторингу розподіленої мережної системи, заснованих на сборі даних щодо фізичного стану хостів та поведінки трафіку, та побудова системи моніторингу мережі на їх основі.

3. Дослідження та оцінка розроблених алгоритмів і їх застосовності в систем моніторингу мережі.
4. Розробка програми, що реалізує функції мережевого моніторингу на основі розробленого методу і алгоритмів.

Перший розділ роботи присвячено аналізу сучасного стану в області моніторингу ВС і розподілених систем, аналізу сучасних методів мережевого моніторингу, а також методів аналізу стека протоколів TCP / IP і їх застосовності в задачах моніторингу мережі.

Другий розділ присвячений розробці методу моніторингу розподіленої мережі та обґрунтуванню вибору параметрів, що підлягають монітронгу.

Третій розділ присвячений розробці програмного забезпечення, що реалізує запропонований метод.

РОЗДІЛ 1.
АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАДАЧІ

Вибір методів і об'єктів моніторингу мережі залежить від безлічі факторів – топології та розміру мережі, діючих в ній сервісів і служб, конфігурації серверів тощо. На базовому рівні найбільш розповсюдженими елементами моніторингу є:

- перевірка фізичної готовності обладнання;
- перевірка стану (працездатності) служб і сервісів, запущених в мережі;
- перевірка завантаженості апаратного блоку системи;
- детальна перевірка параметрів мережі: ефективності функціонування, продуктивності, завантаження тощо;
- перевірка параметрів, специфічних для сервісів і служб даного конкретного оточення (наявність необхідних значень в таблицях БД, вміст лог-файлів).

Початковий рівень будь-якої перевірки передбачає тестування фізичної доступності обладнання. Обладнання може бути не підключене або заблоковані канали зв’язку. Як наслідок – обладнання не готове до використання. Така перевірка передбачає перевірку доступності по ICMP-протоколу (ping), причому зазвичай перевіряється не тільки факт наявності відповіді, але і час проходження сигналу, і кількість загублеих запитів: аномальні значення цих величин, як правило, сигналізують про серйозні проблеми в конфігурації мережі. Деякі з цих проблем легко відстежити за допомогою трасування маршруту (traceroute) – це також можна автоматизувати за наявності «еталонних маршрутів».

Наступний етап передбачає перевірку принципової працездатності критичних служб. Як правило, це означає TCP-підключення до відповідного порту сервера, на якому повинна бути запущена служба, і, можливо, виконання тестового запиту (наприклад, аутентифікації на поштовому сервері за протоколом SMTP або POP або запит тестової сторінки з веб-сервера).

У більшості випадків, бажано перевіряти не тільки факт відповіді служби / сервісу, але і затримки - втім, то відноситься вже до наступної за важливістю завдання: перевірці навантаження. Крім часу відгуку пристроїв і служб для різних типів серверів існують інші принципово важливі перевірки: пам'ять і завантаженість процесора (веб-сервер, сервер БД), місце на диску (файл-сервер), і більш специфічні - наприклад, статус принтерів у сервера друку.

Способи перевірки цих величин варіюються, але один з основних, доступних в більшості випадків - перевірка по SNMP-протоколу. Крім цього, можна використовувати специфічні засоби, що надаються ОС обладнання: наприклад, сучасні серверні версії ОС Windows на системному рівні надають так звані лічильники продуктивності (performance counters), з яких можна "прочитати" досить докладну інформацію про стан комп'ютера.

Нарешті, більшість сервісів оточення вимагають специфічних перевірок - запитів до БД; перевірка файлів звітів ПЗ; відстеження наявності деякого файлу (наприклад, створюваного при «падінні» системи).

1.1. Огляд методів моніторингу мереж

1.1.1. Моніторинг мережевих процесів та стану апаратного блоку

Облік використовуваних апаратних і програмних засобів. Система автоматично збирає інформацію про обстежених комп'ютерах і створює записи в базі даних про апаратних і програмних ресурсах. Після цього адміністратор може швидко з'ясувати, що в нього є і де це знаходиться. Наприклад, дізнатися про те, на яких комп'ютерах потрібно оновити драйвери принтерів, які ПК володіють достатньою кількістю пам'яті і дискового простору тощо.

Розподіл і установка програмного забезпечення. Після завершення обстеження адміністратор може створити пакети розсилки програмного забезпечення - дуже ефективний спосіб для зменшення вартості такої процедури. Система може також дозволяти централізовано встановлювати і адмініструвати додатки, які запускаються

з файлових серверів, а також дати можливість кінцевим користувачам запускати такі додатки з будь-якої робочої станції мережі[3].

Віддалений аналіз продуктивності і виникаючих проблем. Адміністратор може дистанційно керувати мишею, клавіатурою і бачити екран будь-якого ПК, що працює в мережі під управлінням тієї чи іншої мережної операційної системи. База даних системи управління зазвичай зберігає детальну інформацію про конфігурацію всіх комп'ютерів в мережі для того, щоб можна було виконувати віддалений аналіз виникаючих проблем.

Прикладами засобів управління розподіленими системами можуть слугувати такі продукти, як LANDeskManager фірми або Intel SystemManagementServer компанії Microsoft, а типовими представниками засобів управління мережами є системи HPOpenView, SunNetManager і IBMNetView.

В області систем управління останнім часом спостерігаються такі чітко виражені тенденції:

- інтеграція в одному продукті функцій управління мережами і системами,
- розподіленість системи управління, при якій в системі існує кілька агентів, які збирають інформацію про стан системи і видають керуючі сигнали.

Створення систем управління мережами неможливе без орієнтації на певні стандарти, так як управляюче програмне забезпечення та мережеве обладнання, а, також, і агентів для нього, розробляють сотні компаній. Оскільки корпоративні мережі переважно неоднорідні, керуючі інструменти не можуть відображати специфіку однієї системи або мережі[1].

Найбільш поширеним протоколом управління мережами є протокол SNMP (SimpleNetworkManagementProtocol), його підтримують сотні виробників. Головні переваги протоколу SNMP - простота, доступність, незалежність від виробників. Значною мірою саме популярність SNMP затримала прийняття CMIP, варіанти керуючого протоколу за версією OSI. Протокол SNMP розроблений для управління маршрутизаторами в мережі Internet і є частиною стека TCP / IP.

SNMP - це протокол, який застовується для отримання від мережевих пристроїв інформації про їх стан, продуктивність і характеристики, які зберігаються

в спеціальній базі даних мережевих пристроїв, яка називається MIB (ManagementInformationBase)[2]. Існують стандарти, що визначають структуру MIB, в тому числі набір типів її змінних (об'єктів в термінології ISO), їх імена і допустимі права доступу. У MIB, поряд з іншою інформацією, можуть зберігатися мережевий і / або MAC-адреси пристроїв, значення лічильників оброблених пакетів і помилок, номери, пріоритети та інформація про стан портів. MIB має деревоподібну будову. Така структура містить обов'язкові (стандартні) піддерева, а також в ній можуть знаходитися приватні (private) піддерева.

Агент в протоколі SNMP - це програмний елемент, який забезпечує менеджерам, розміщеним на керуючих станціях мережі, доступ до значень змінних MIB, і тим самим дає їм можливість реалізовувати функції з управління та нагляду за пристроєм[2].

Основні операції з управління винесені на керуючу станцію. При цьому пристрій працює з мінімальними витратами на підтримку керуючого протоколу. Він використовує майже всю свою обчислювальну потужність для виконання своїх основних функцій маршрутизатора, моста або концентратора. Програмний агент займається збором статистики і значень змінних стану пристрою і передачею їх менеджеру системи управління. SNMP - це протокол типу "запит-відповідь". Отже, на кожен запит, що надійшов від менеджера, агент повинен передати відповідь. Особливістю протоколу є його надзвичайна простота - він включає в себе лише кілька команд[].

Команда Get-request використовується менеджером для отримання від агента значення будь-якого об'єкта за його ім’ям.

Команда GetNext-request використовується менеджером для вилучення значення наступного об'єкта (без зазначення його імені) при послідовному перегляді таблиці об'єктів.

За допомогою команди Get-response агент SNMP передає менеджеру відповідь на одну з команд Get-request або GetNext-request.

Команда Trap використовується агентом для повідомлення менеджеру про виникнення особливої ситуації.

Команда Set використовується менеджером для встановлення значення будь-якого об'єкта або умови, при виконанні якого агент SNMP повинен послати менеджеру відповідне повідомлення. Може бути визначена відповідна дія на такі події як ініціалізація агента, рестарт агента, невірна аутентифікація, обрив зв'язку, відновлення зв'язку і втрата найближчого маршрутизатора. Якщо відбувається будь-яка з цих подій, то агент ініціалізує переривання[1].

Версія SNMPv.2 додає до цього переліку команду GetBulk, яка дозволяє менеджеру отримати кілька значень змінних за один запит.

1.1.2. Інтегровані системи управління та аналізу

Агенти SNMP. На сьогодні існує кілька стандартів на бази даних керуючої інформації. Основними є стандарти MIB-I і MIB-II, а також версія бази даних для віддаленого управління RMONMIB. Окрім цього, існують стандарти для спеціальних MIB пристроїв конкретного типу (наприклад, MIB для концентраторів або MIB для модемів), а також приватні MIB конкретних фірм-виробників обладнання[1].

Базова специфікація MIB-I визначала тільки операції читання значень змінних. Операції зміни або установки значень об'єкта є частиною специфікацій MIB-II.

Версія MIB-I (RFC 1156) визначає до 114 об'єктів, які поділяються на 8 груп[5]:

- System - загальні дані про фізичний пристрій (як то, ідентифікатор постачальника, час останньої ініціалізації системи тощо).
- Interfaces – надає значення параметрів мережевих інтерфейсів пристрою (наприклад, їх кількість, типи, швидкості обміну, максимальний розмір пакета).
- AddressTranslationTable – надає інформацію про відповідність між мережевими і фізичними адресами (наприклад, за протоколом ARP).

- InternetProtocol - дані, що відносяться до протоколу IP (адреси IP-шлюзів, хостів, статистика про IP-пакетах).
- ICMP - дані, що відносяться до протоколу обміну керуючими повідомленнями ICMP.
- TCP - дані, що відносяться до протоколу TCP (наприклад, про TCP-судинних).
- UDP - дані, що відносяться до протоколу UDP (число переданих, прийнятих і помилкових UPD-дейтаграмм).
- EGP - дані, що відносяться до протоколу обміну маршрутною інформацією ExteriorGatewayProtocol, використовуваному в мережі Internet (число прийнятих з помилками і без помилок повідомлень).

З цього переліку груп змінних видно, що стандарт MIB-I розроблявся виключно орієнтовано на управління маршрутизаторами, що підтримують протоколи стека TCP / IP. У версії MIB-II (RFC 1213), прийнятої в 1992 році, був істотно (до 185) розширено набір стандартних об'єктів, а кількість груп збільшилася до 10.

Агенти RMON. Специфікація RMON є розширенням до функціональних можливостей SNMP, яка забезпечує віддалену взаємодію з базою MIB. До запровадження RMON протокол SNMP не міг використовуватися віддалено, він допускав лише локальне управління пристроями. База RMONMIB володіє поліпшеним набором властивостей для віддаленого управління, так як містить агреговану інформацію про пристрій, що не вимагає передачі по мережі великих обсягів інформації. Об'єкти RMONMIB включають додаткові лічильники помилок в пакетах, гнучкіші засоби аналізу графічних трендів і статистики, більш потужні засоби фільтрації для захоплення і аналізу окремих пакетів, а також більш складні умови встановлення сигналів попередження. Агенти RMONMIB більш інтелектуальні порівняно з агентами MIB-I або MIB-II і виконують значну частину роботи по обробці інформації про пристрій, яку раніше виконували менеджери. Ці агенти можуть розташовуватися всередині різних комунікаційних пристроїв, а

також бути виконані у вигляді окремих програмних модулів, що працюють на універсальних ПК і ноутбуках (прикладом може служити LANalyzerNovell)[2].

Об'єкту RMON присвоєно номер 16 в наборі об'єктів MIB, а сам об'єкт RMON об'єднує 10 груп таких об'єктів[5]:

- Statistics - поточні накопичені статистичні дані про характеристики пакетів, кількості колізій тощо.
- History - статистичні дані, збережені через певні проміжки часу для подальшого аналізу тенденцій їх змін.
- Alarms - порогові значення статистичних показників, при перевищенні яких агент RMON посилає повідомлення менеджеру.
- Host - даних про хостах мережі, в тому числі і про їх MAC-адресах.
- HostTopN - таблиця найбільш завантажених хостів мережі.
- TrafficMatrix - статистика про інтенсивність трафіку між кожною парою хостів мережі, впорядкована у вигляді матриці.
- Filter - умови фільтрації пакетів.
- PacketCapture - умови захоплення пакетів.
- Event - умови реєстрації і генерації подій.

Дані групи пронумеровані в зазначеному порядку, тому, наприклад, група Hosts має числове ім'я 1.3.6.1.2.1.16.4.

Десяту групу складають спеціальні об'єкти протоколу TokenRing.

Всього стандарт RMONMIB визначає близько 200 об'єктів в 10 групах, зафіксованих в двох документах - RFC 1271 для мереж Ethernet і RFC 1513 для мереж TokenRing.

Відмінною рисою стандарту RMONMIB є його незалежність від протоколу мережевого рівня (на відміну від стандартів MIB-I і MIB-II, орієнтованих на протоколи TCP / IP). Тому, його зручно використовувати в гетерогенних середовищах, що використовують різні протоколи мережевого рівня.

1.1.3. Спеціалізовані методи моніторингу

Аналізатори протоколів. Одним з найбільш досконалих та розповсюджених засобів дослідження мережі є аналізатори протоколів. Процес аналізу протоколів включає захоплення циркулюючих в мережі пакетів, що реалізують той чи інший мережевий протокол, і вивчення вмісту цих пакетів. Ґрунтуючись на результатах аналізу, можна здійснювати обґрунтовану і зважену оцінку будь-яких компонент мережі, оптимізацію продуктивності мережі, пошук і усунення відмов. Очевидно, що для того, щоб можна було зробити якісь висновки про вплив деяких змін на мережу, необхідно виконати аналіз протоколів на достатньому проміжку часу роботи мережі[17].

Аналізатор протоколів є або самостійним спеціалізованим пристроєм, або персональним комп'ютером, який оснащений спеціальною мережевою картою і відповідним програмним забезпеченням. Застосовувані мережева карта і програмне забезпечення повинні відповідати топології мережі. Аналізатор підключається до мережі як звичайний вузол. Відмінність полягає в тому, що аналізатор може приймати всі пакети даних, що передаються по мережі, в той час як звичайна станція - лише пакети адресовані їй.

Програмне забезпечення аналізатора складається з ядра, що підтримує роботу мережного адаптера і декодера одержуваних даних, і додаткового ПЗ, що залежить від типу топології досліджуваної мережі. Крім того, поставляється ряд процедур декодування, орієнтованих на певний протокол, наприклад, IPX. До складу деяких аналізаторів може входити також експертна система, яка може видавати користувачу рекомендації про те, які дії слід проводити в даній ситуації; що можуть означати ті чи інші результати вимірювань; як усунути деякі види несправності мережі.

Незважаючи на відносне різноманіття аналізаторів протоколів, представлених на ринку, можна визначити деякі риси, в тій чи іншій мірі властиві всім їм[8]:

- Користувальницький інтерфейс. Більшість аналізаторів мають розвинений дружній інтерфейс, який базується, як правило, на Windows або Motif. Цей інтерфейс дозволяє користувачеві: виводити результати аналізу інтенсивності трафіку; отримувати миттєву і усереднену статистичну оцінку продуктивності

мережі; моделювати певні події і критичні ситуації для відстеження їх виникнення; декодувати протоколи різного рівня і надавати в зручній для аналізу формі вміст пакетів.

- Буфер захоплення. Буфери різних аналізаторів відрізняються за об’ємом. Буфер може розташовуватися на встановленій мережевій карті, або для нього може бути відведено місце в оперативній пам'яті одного з комп'ютерів мережі. Якщо буфер розташований на мережевій карті, то управління ним здійснюється апаратно, і за рахунок цього швидкість введення зростає. Однак, це призводить до підвищення ціни аналізатора. У разі недостатньої продуктивності процедури захоплення, частина інформації буде губитися, і аналіз буде неможливий. Обсяг буфера визначає можливості аналізу по більш-менш представницьким вибірках захоплюваних даних. Але яким би великим не був буфер захоплення, рано чи пізно він заповниться. У цьому випадку або припиняється захоплення, або заповнення починається з початку буфера.
- Фільтри. Фільтрація інформації дозволяє управляти процесом захоплення даних, і, тим самим, надає можливість економити простір буфера. Залежно від значення певних полів пакету, заданих у вигляді умови фільтрації, пакет або ігнорується, або записується в буфер захоплення. Використання фільтрів значно прискорює і спрощує аналіз, так як виключає перегляд непотрібних в даний момент пакетів.
- Перемикачі - задаються оператором як деякі умови початку і припинення процесу захоплення даних з мережі. Такими умовами можуть бути виконання ручних команд запуску і зупинки процесу захоплення, час доби, тривалість процесу захоплення, поява певних значень в кадрах даних. Перемикачі можуть використовуватися одночасно з фільтрами, дозволяючи більш детально і точно проводити аналіз, а також ефективніше використовувати обмежений обсяг буфера захоплення.
- Пошук. Деякі аналізатори протоколів передбачають можливість пошуку потрібної інформаці що знаходиться в буфері. Це дозволяє автоматизувати перегляд змісту пакетів, і знаходити в них дані за заданими критеріями. У той час, як фільтри

перевіряютьвхідний потік на предмет відповідності умовам фільтрації, функції пошуку застосовуються до вже накопиченим в буфері даними.

Методологія проведення аналізу може бути надана у вигляді наступних шести етапів[5]:

1. Відстеження даних.
2. Перегляд захоплених даних.
3. Дослідження інформації.
4. Відстеження помилок. (Більшість аналізаторів надають можливість визначаючи типи помилок. а також, ідентифікувати станцію, від якої прийшов пакет з помилкою.)
5. Аналіз продуктивності. Розраховується коефіцієнт використання пропускної здатності мережі або середній час реакції на запит.
6. Ретельне дослідження окремих ділянок мережі. Зміст цього етапу конкретизується у міру того, як проводиться аналіз.

Зазвичай процес аналізу протоколів займає відносно небагато часу - 1-2 робочих дня.

Сертифікація и тестування кабельних мереж. До обладнання даного класу відносяться мережеві аналізатори, пристрої для сертифікації кабелів, кабельні сканери і тестери. Перш, ніж перейти до більш докладного розгляду цих пристроїв, розглянемо деякі необхідні відомості про основні електромагнітні характеристики кабельних систем.

Основними електричними характеристиками, що впливають на роботу кабелю, є: загасання, імпеданс (хвильовий опір), перехресні наведення двох кручених пар і рівень зовнішнього електромагнітного випромінювання.

Мережеві аналізатори. Мережеві аналізатори (не слід плутати їх з аналізаторами протоколів) представляють собою еталонні вимірювальні інструменти для діагностики і сертифікації кабелів і кабельних систем.

Мережеві аналізатори містять високоточний частотний генератор і вузькосмуговий приймач. Передаючи сигнали різних частот в передавальну пару і

вимірюючи сигнал в приймальній парі, можна виміряти затухання і NEXT. Мережеві аналізатори - це прецизійні великогабаритні і дорогі прилади, призначені для використання спеціально навченим технічним персоналом[1].

Кабельні сканери. Дані прилади дозволяють визначити довжину кабелю, NEXT, загасання, схему розведення, імпеданс, рівень електричних шумів і провести оцінку отриманих результатів. Ціна на ці прилади варіюється від $ 1'000 до $ 3'000. Існує досить багато пристроїв даного класу, наприклад, сканери компаній MicrotestInc., FlukeCorp., DatacomTechnologiesInc., ScopeCommunicationInc. На відміну від мережевих аналізаторів сканери можуть бути використані не тільки спеціально навченим технічним персоналом, але навіть адміністраторами-новачками.

Для визначення місця та причини несправності кабельної системи (обриву, короткого замикання, невірно встановленого роз'єму і т.д.) застовується метод "кабельного радара", або Time Domain Reflectometry (TDR). Суть цього методу полягає в тому, що сканер випромінює в кабель короткий електричний імпульс і вимірює час затримки до приходу відбитого сигналу. За полярності відображеного імпульсу визначається характер пошкодження кабелю (коротке замикання або обрив). В правильно встановленому і підключеному кабелі акустичний імпульс зовсім відсутній.

Точність визначення відстані залежить від того, наскільки точно відома швидкість поширення електромагнітних хвиль у кабелі. У різних кабелях вона буде різною. Швидкість поширення електромагнітних хвиль у кабелі (NVP – nomina lvelocityof propagation) звичайно задається у відсотках до швидкості поширення світла у вакуумі. Сучасні сканери містять електронну таблицю даних про NVP для всіх основних типів кабелів. Це дозволяє користувачеві встановлювати ці параметри самостійно після попереднього калібрування.

Найбільш відомими виробниками компактних (їх розміри звичайно не перевищують розміри відеокасети стандарту VHS) кабельних сканерів є компанії MicrotestInc., WaveTekCorp., ScopeCommunicationInc[1].

Тестери кабельних систем - найбільш прості і дешеві прилади для діагностики кабелю. Вони дозволяють визначити безперервність кабелю, однак, на відміну від кабельних сканерів, не дають відповіді на питання про те, в якому місці стався збій.

1.2. Огляд інструментів моніторингу стану розподілених систем

Моніторинг - (відстеження стану сервісів), поряд з резервним копіюванням є, напевно, однією з найстаріших і найпопулярніших функцій системного адміністратора. Отже, існує безліч різних інструментів для її виконання. В даному розділі зібрано основні системи моніторингу з коротким описом і порівняльним аналізом, відзначенням плюсів, мінусів і особливостей.

Monit Одна з перших розробок. Офіційна назва - That barking on daemons. Працює строго локально, самостійний демон, написаний на мові C. Не має можливості розширення - не підтримує плагіни. Основне завдання - відслідковувати роботу демонів і перезапускати померлі, які зависли або вийшли за квоту ресурсів. Має своєрідну конфігурацію. Не дивлячись на нульову розширюваність в базі має більшу частину необхідних перевірок. Може перевіряти процес за фактом наявності, займані ресурси, підключатися до процесу (по мережі або сокету), перевіряти відповідь від сервера (на факт наявності, а так само на утримання), перевіряти відповідність певним протоколам (HTTP, FTP, POP / IMAP / SMTP, тощо. Підтримує роботу з SSL, має вбудований веб-інтерфейс. Перевірка проводиться за розкладом, через задані проміжки часу. За результатами перевірки може приймати певні рішення (зупинити процес, перезапустити, повідомити адміністратора мережі). В активі компанії-учасника є многосервісна платна система M / Monit - для управління групами серверов.

Munin Цей проект створювався спочатку для малювання графіків, він може служити і моніторинговою системою (так, як має в базі систему повідомлень). Строго клієнт-серверна система, на цільових вузлах запускається процес-клієнт. Процес-сервер підключається до клієнтів за розкладом і збирає числові дані (метрики), зберігає їх у себе і вимальовує графіки. Він же відповідає за

повідомлення. Munin легко розширюється: всі метрики малює, використовуючи дані плагінів (без плагінів метрик не буде). Модулі легко пишуться - мова довільна, так як Munin очікує тільки числа (дані) метрик в певному форматі і код відповіді. В силу своєї архітектури Munin не здатний виконувати дій на клієнтських вузлах (наприклад - перезапускати завислий процес) [17].

Ganglia Досить старий інструмент, створений в CERN. Основне призначення ганглії - збір даних про продуктивність великої кількості однотипних машин (обчислювальні кластери). Має продуктивність, вражає уяву (десятки тисяч спостережуваних вузлів на дуже середньому сервері), але трешхолд досить великий (дані збираються відкладення, так що якщо треба дивитися навантаження в реальному часі - це не до нього. Клієнт-серверна архітектура: клієнт збирає дані, акумулює їх у себе і раз в певний час відсилає на сервер. у разі, якщо якась метрика переступила порогове значення, відповідь буде відправлений позачергово. сервер збирає дані з усіх клієнтів, відредагує їх і зберігає в базі. Веб- інтерфейс є окремим компонентом, він зв'язується з сервером по внутрішньому протоколу і відображає різні графіки. В силу специфіки ганглії інтерфейс дуже незвичайний, але прекрасно підходить для порівняльного аналізу даних з груп серверів (наприклад, побудова графіка залежності навантаження на CPU сервера 10 від мережевого навантаження на сервери з 50 по 90 робиться в два кліка мишею). Ганглія розширювана, але написання плагіна представляє з себе нетривіальну задачу - справа в тому, що з точки зору ганглії, плагін - це бібліотека. Це накладає вимоги на мову реалізації плагіна (тільки C або Python) і внутрішню структуру коду (описано в документації). Самі демони написані на C, веб-інтерфейс - на PHP. Ганглія має достатню систему повідомлень, найчастіше ганглію компонують з Ісінгой . В силу специфіки ганглії інтерфейс дуже незвичайний, але прекрасно підходить для порівняльного аналізу даних з груп серверів (наприклад, побудова графіка залежності навантаження на CPU сервера 10 від мережевого навантаження на сервери з 50 по 90 робиться в два кліка мишею). В силу специфіки ганглії інтерфейс дуже незвичайний, але прекрасно підходить для порівняльного аналізу даних з груп серверів (наприклад, побудова

графіка залежності навантаження на CPU сервера 10 від мережевого навантаження на сервери з 50 по 90 робиться в два кліка мишею). [16]

Nagios / Icinga

Досить старий Інстурмент, спочатку називався Nagios (автор - Етан Галстадт). В процесі розробки з'явився форк (Icinga). Зараз ці системи розвиваються самостійно, але загального в них набагато більше, ніж різниці. Nagios існує в безкоштовній (core) і платній редакціях, Icinga - Open source. Спочатку серверно-орієнтований демон на мові C, управління демоном ведеться через C-подібний конфиг (подібний до конфиг ISC BIND). Всі перевірки збираються в одну чергу і виконуються в чіткій послідовності. Це накладає певні обмеження на масштабованість - сервер виконує велику кількість перевірок, але чим їх більше, тим більше часу проходить між ними. Пара тисяч перевірок (400 серверів х 5 перевірок на сервер) робить систему моніторингу не ефективною - між перевірками проходить занадто багато часу. В якості опції існує додаток-клієнт під назвою NRPE, він дозволяє проводити перевірку локально, на клієнті (для тих випадків, коли перевірити сервіс віддалено неможливо), але це саме опція. Крім того, перевірки через клієнта проводяться з ініціативи сервера, так що на масштабованість це ніяк не впливає. Nagios дуже добре розширюється, плагіни пишуться дуже просто їх існує безліч. Nagios відрізняється потужною і гнучкою системою повідомлень - в базі вона вміє писати листи і слати смс і повідомлення на пейджер. Є цілий набір плагінів - від автоматичного дзвінка голосом до світлофора. Управляється ця система через ModBus. Icinga має два веб-інтерфейси на вибір - класичний CGI і сучасніший на PHP. Суб'єктивно CGI зручніше, хоча і менш гнучкий. Попередня версія (nagios core) має тільки CGI-версію інтерфейсу. Nagios не має функцій збору та аналізу інформації. А отже, не надає статистику відповідей. Існує міст для зв'язку Icinga з Ganglia. В цьому дуеті Icinga відповідає, в першу чергу, за карту мережі, огляд хостів і повідомлення. [19]

Cacti Спочатку цей інструмент був призначений для швидкої і простої збірки статистики по SNMP, але, з застосуванням плагінів (notify + treshold) він набув можливості системи моніторінга. Це PHP-додаток, налаштувань і конфігурації

зберігаються в базі MySQL (статистика зберігається в RRD, що підвищує продуктивність сервісу). Дуже проста установка і базове налаштування. Все відбувається через Інтернет. Програма непогано масштабується, завдяки застосованню многопоточного опитувальника spine – SNMP. Має велику кількість базових шаблонів для перевірки різних типів пристроїв. Основний мінус програми - в її SNMP-центричності. Даний сервіс добре підходить для збору даних з активного обладнання, середньо - для збору типової статистики роботи сервісу (дисковий простір, завантаження CPU etc) і не ефективна - для збору складних і нетипових метрик. Кастомізація майже відсутня.[17]

OpenNMS Це складна, многокомпонентна система, написаний на Java. Має власний app-server, орієнтований на великі інфраструктури з передачею статистики по SNMP. Велика складність і гнучкість сервісу роблять налаштування досить складним. Для невеликих інфраструктур цей сервіс надмірний. Так само, як і cacti, сервіс розроблений для роботи з SNMP, а отже, основне застосування сервісу - це моніторинг мережевих пристроїв. Сервіс страждає практично повною відсутністю документації і дуже погано розширюється (по суті це монолітний java-додаток, внутрішні налаштування існують у вигляді XML-файлів, що дуже ускладнює налаштування цієї програми..

Graphite + CollectD + Whisper Кожен компонент цього набору сам по собі не вирішує проблему моніторингу. Якщо всі вище перераховані продукти - це готові рішення. Незручні файли конфігурації, низька стабільність коду (версії виду 0.1.192-alpha) і повна відсутність документації компенсується швидкістю роботи і масштабованість. Ці програми надають можливість отримувати дані з серверної ферми класу github і бачити дані в практично реальному часі.

Zabbix Автори цього інструменту при розробці брали прикладом серйозні системи моніторингу enterprice-класу. По суті - zabbix - це моніторинг enterprice-класу, але при цьому він opensource. Володіє триланковою архітектурою (сервер-проксі-клієнт, проксі опціональний, але допомагає знизити навантаження), дані зберігає в SQL, веб-інтерфейс - повністю самостійне додаток на php. Самі демони моніторингу написані на C / C ++. Всі налаштування так само зберігаються в базі

даних, а змінюються через веб. Система володіє на рідкість розлогим функціоналом. Безліч різних перевірок, графіки "від чого завгодно і куди завгодно", ескалації проблем і відстеження SLA, app-level перевірки, імітація ходіння користувача по сайту (з підтримкою javascript, cookies, GET / POST / PUT), карти і схеми, настроюються панелі (dashboards) з будь-якими метриками на вибір. Система має свій API, розподілену модель прав доступу, має кластерну структуру для зниження навантаження на вузли, має шаблони перевірок. На відміну від більшості перерахованих вище систем, zabbix може виробляти дії на клієнтів (наприклад - перезапустити певний процес). Система чудово розширюється і дуже гнучко налаштвується. В SQL заббікс зберігаються абсолютно всі - налаштування, статистика, метрики, вузли. Активно працює zabbix виїдає IOPS з великою швидкістю. Цьому може запобігти зниження кількість метрик і частоти їх збору, але це знижує користь самої системи моніторингу. В такому разі, частина інформації губиться. Zabbix уадзвичайно гнучка по суті, система дуже непроста в установці (навіть з урахуванням того, що у неї чудово написана документація, в ній висвітлені практично всі питання). Впровадження заббікса з нуля вільно може зайняти місяць праці адміністратора, а оптимізація під завдання бізнесу ще довше. [18]

Висновки до розділу 1

Моніторинг - це збір, обробка, агрегування і відображення кількісних даних в реальному часі про систему, таких як кількість запитів і типи запитів, кількість і типи помилок, час обробки, час життя і ресурсо-вживання сервера / клієнта / додатки / сервісу.

Існує безліч різних інструментів для здійснення моніторингу. Найпопулярніші з них – Zabbix, OpenNMS, Cacti, Nagios є системами це моніторингу enterprice-класу. Вони доволі громіздкі, вимагають високої кваліфікації від операторів та спеціалізуються, здебільшого на аналізі мережевого трафіку та стану комунікаційного обладнання.

Моніторинг мереж можна розподілити на наступні складові:

- тестування фізичної доступності обладнання;
- перевірка принципової працездатності критичних служб;
- перевірка завантаженості трафіку;
- перевірка завантаженості апаратної частини системи.

Використання систем моніторингу та управління IT-інфраструктурою дозволяє:

- перевірити фізичну доступність обладнання та оптимізувати використання інформаційних ресурсів;
- підвищити якість IT-сервісів і швидкість усунення збоїв в роботі обладнання і програмного забезпечення;
- забезпечити надійність, безпеку і узгоджене функціонування всіх компонентів IT-інфраструктури;
- полегшити модернізацію IT-інфраструктури;
- в кілька разів підвищити ефективність роботи системних адміністраторів.

РОЗДІЛ 2.
РОЗРОБКА МЕТОДУ МОНІТОРИНГУ СТАНУ АПАРАТНИХ РЕСУРСІВ МЕРЕЖІ

2.1. Особливості структури розподіленої мережної системи

Одним з найбільш поширених прикладів використання розподіленої мережної архітектури є розподілені системи управління (РСУ).

Розподілену систему управління (РСУ) можна визначити як систему, що складається з безлічі пристроїв, розташованих в просторі, кожне з яких не залежить від інших, але взаємодіє з ними для виконання спільного завдання. Як "безліч пристроїв" можуть виступати будь-які мікропроцесорні пристрої.(рис. 2.1)

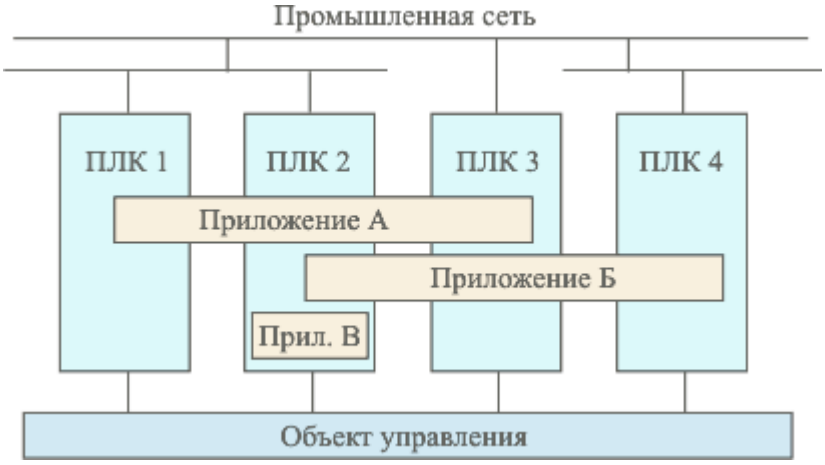

Рис. 2.1. Модель рзподіленої СА у в відповідності до стандарту МЕК 61499

Модель розподіленої системи автоматизації відповідно до стандарту МЕК 61499 може бути представлена як набір фізичних пристроїв взаємодіючих між собою за допомогою однієї або декількох промислових мереж. Посилання на МЕК 61499

В об'єднаних мережах TCP / IP управління мережею будується на взаємодії між менеджером (станцією управління мережею) і елементами мережі. Станції

управління це звичайні робочі станції з графічним дисплеєм і кольоровим монітором, які відображають все, що відбувається з елементами (які з них працюють, а які ні, обсяг трафіку по різних каналах за одиницю часу і так далі. Елементами мережі можуть бути: хости, маршрутизатори, X термінали, термінальні сервера, принтери і так далі. Тобто, будь-які об'єкти, які використовують сімейство протоколів TCP / IP. Всі елементи мережі повинні працювати із програмним забезпеченням, яке називається агентом..[3]

Агент є програма посередник між керованим ресурсом і головною управляючою програмою-менеджером. Щоб один і той же менеджер міг управляти різними реальними ресурсами, створюється деяка модель керованого ресурсу, яка відображає тільки ті параметри ресурсу, які потрібні для його управління і контролю. Прикладом може слугувати модель маршрутизатора, яка зазвичай включає такі характеристики, як кількість портів, їх тип, таблицю маршрутизації, кількість кадрів і пакетів протоколів канального, мережевого і транспортного рівнів, які пройшли через ці порти.

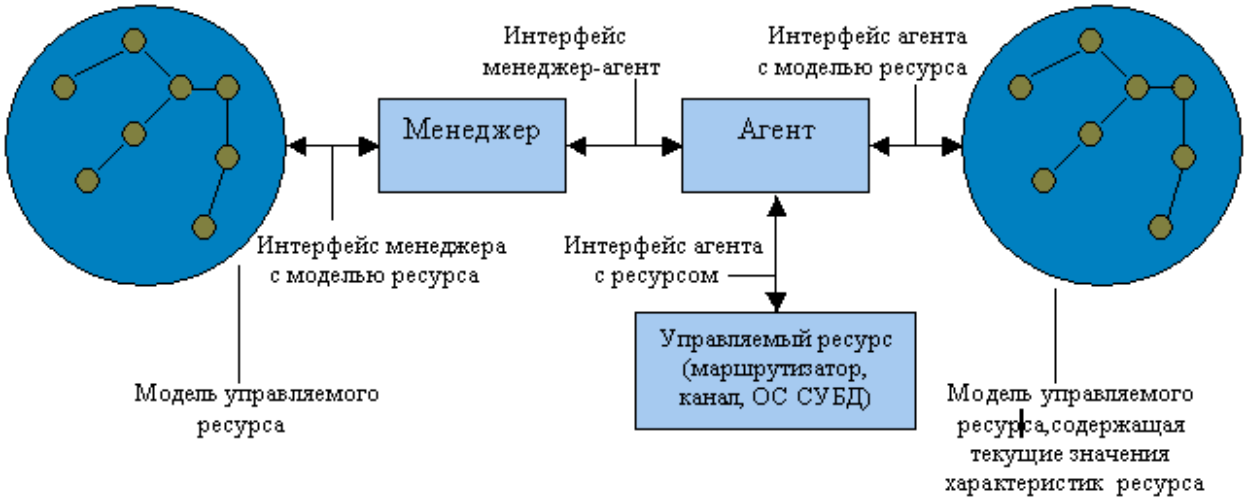

Рис.2.2. Взаємодія агента, менеджера і керованого ресурса

Агент є деяким екраном, що звільняє менеджера від непотрібної інформації про деталі реалізації ресурсу Менеджер отримує від агента тільки ті дані, які описуються моделлю ресурсу.. Агент надає менеджеру оброблену і подану в формалізованому вигляді інформацію. Базуючись на цій інформації менеджер

приймає рішення по управлінню, а також виконує подальше узагальнення даних про стан керованого ресурсу. Наприклад, досліджує залежність завантаження порту від часу. Для отримання необхідних даних від об'єкта, а також для видачі на нього керуючих впливів агент взаємодіє з реальним ресурсом деяким нестандартним способом. В процесі встановлення агента в комунікаційне обладнання, розробник обладнання передбачає точки і способи взаємодії внутрішніх вузлів пристрою з агентом. При розробці агента для певної операційної системи розробник агента користується тими інтерфейсами, які існують в цій ОС, як то: інтерфейсами ядра, драйверів і додатків. Агент може забезпечуватися спеціальними засобами для отримання інформації, наприклад датчиками температури або датчиками релейних контактів. (рис.2.2.)

Менеджер і агент повинні мати у своєму розпорядженні одні і ті ж моделі керованого ресурсу, інакше вони не зможуть співпрацювати одне з одним. Однак цієї моделі у відношенні з агентом і менеджером зовсім різні. Менеджер використовує модель, щоб отримати відомості про те, чим характеризується ресурс, які характеристики він може вимагати від агента і якими параметрами можна керувати. Менеджер взаємодіє з агентами за стандартним протоколом. Цей протокол повинен дозволяти менеджеру запитувати значення параметрів, що зберігаються в базі MIB, а також передавати агенту керуючу інформацію, на основі якої той повинен управляти пристроєм. Агент наповнює модель керованого ресурсу поточними значеннями характеристик даного ресурсу, і в зв'язку з цим модель агента називають базою даних керуючої інформації - Management Information Base, MIB. [6]

За шляхами передачі інформації розрізняють управління in-band, тобто по тому ж каналу, по якому передаються призначені для користувача дані, і управління out-of-band, тобто поза каналу, по якому передаються призначені для користувача дані. Наприклад, якщо менеджер взаємодіє із агентом, вбудованим в маршрутизатор, по протоколу SNMP, переданому по тій же локальній мережі, що і призначені для користувача дані, то це буде управління in-band. Якщо ж менеджер контролює комутатор первинної мережі, що працює за технологією частотного ущільнення

FDM, за допомогою окремої мережі X.25, до якої підключений агент, то це буде управління out-of-band. Здійснення керування з того ж каналу, по якому працює мережа, більш економічно, так як не вимагає створення окремої інфраструктури передачі керуючих даних. Однак спосіб out-of-band значно надійніший, тому що він надає можливість керувати устаткування мережі і тоді, коли деякі елементи мережі вийшли з ладу і по основних каналах обладнанням недоступне. Стандарт багаторівневої системи управління TMN має в своїй назві слово Network, що підкреслює, що в загальному випадку для управління телекомунікаційною мережею створюється самостійна керуюча мережа, яка забезпечує режим out-of-band.

Звичайно менеджер працює з кількома агентами, обробляючи отримані від них дані і видаючи на них управлінські команди. Агенти можуть встановлюватися в кероване обладнання, також можуть працювати на окремому комп'ютері, пов'язаному з керованим обладнанням з деякого інтерфейсу. Звичайно менеджер працює на окремому комп'ютері, який одночасно виконує роль консолі управління для адміністратора або оператора системи.

Модель менеджер - агент є базовою основою дляі таких популярних стандартів управління, як стандарти Internet на основі протоколу SNMP і стандарти управління ISO / OSI на основі протоколу CMIP. Агенти можуть відрізнятися рівнем інтелекту - вони можуть володіти як зовсім мінімальним інтелектом, необхідним для підрахунку кадрів і пакетів, що проходять через обладнання, так і вельми високим, достатнім для виконання самостійних дій по виконанню послідовності керуючих дій в аварійних ситуаціях, фільтрації аварійних повідомлень, побудови часових залежностей і т.ін.

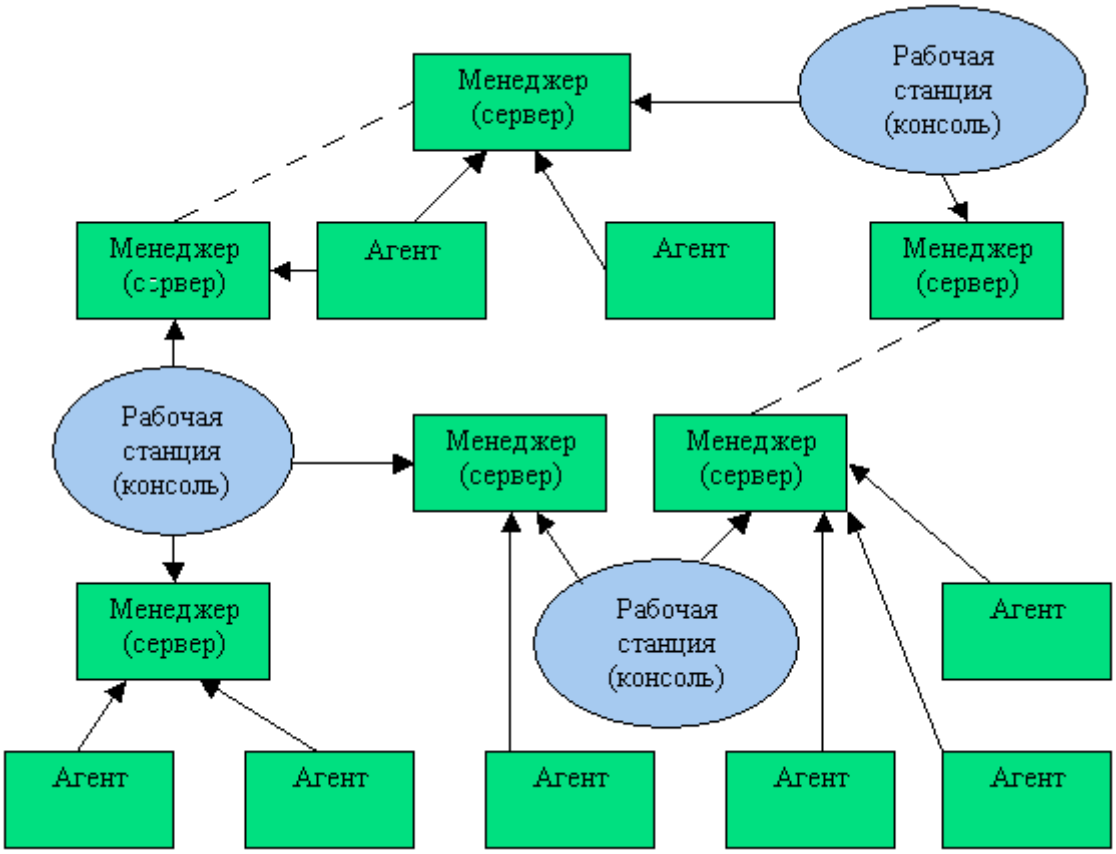

Рис. 2.3. Розподілена система управління на основі декількох менеджерів і робочих станцій

Схема менеджер - агент дозволяє будувати досить складні структури розподіленої системи управління.(рис. 2.3.)

Звичайно, розподілена система управління істить велику кількість зв'язків менеджер - агент, які доповнюються робочими станціями операторів мережі, за допомогою яких вони отримують доступ до менеджерів (рис. 2.). Кожен агент збирає дані і управляє певним елементом мережі. Менеджери, іноді також являються серверами системи управління, отримують дані від своїх агентів, аналізують, узагальнюють їх і зберігають в базі даних. Оператори, що працюють за робочими станціями, можуть з'єднатися з кожним з менеджерів і за допомогою графічного інтерфейсу переглянути дані про керовану мережу, а також видати менеджеру деякі директиви з управління мережею або її елементами.[9]

Набагато більш гнучкою є ієрархічна побудова зв'язків між менеджерами. Кожен менеджер нижнього рівня виконує також функції агента для менеджера верхнього рівня. Такий агент працює вже з набагато більш вдосконаленою моделлю

(MIB) своєї частини мережі, в якій збирається саме таку інформацію, яка потрібна менеджеру верхнього рівня для управління мережею взагалі. Звичайно для розробки моделей мережі на різних рівнях проектування починають з верхнього рівня, на якому визначається склад інформації, яка вимагається від менеджерів-агентів нижчого рівня, тому такий підхід названий підходом «зверху вниз». Такий підхід скорочує обсяги інформації, що циркулює між рівнями системи управління, і призводить до набагато більш ефективної системи управління.

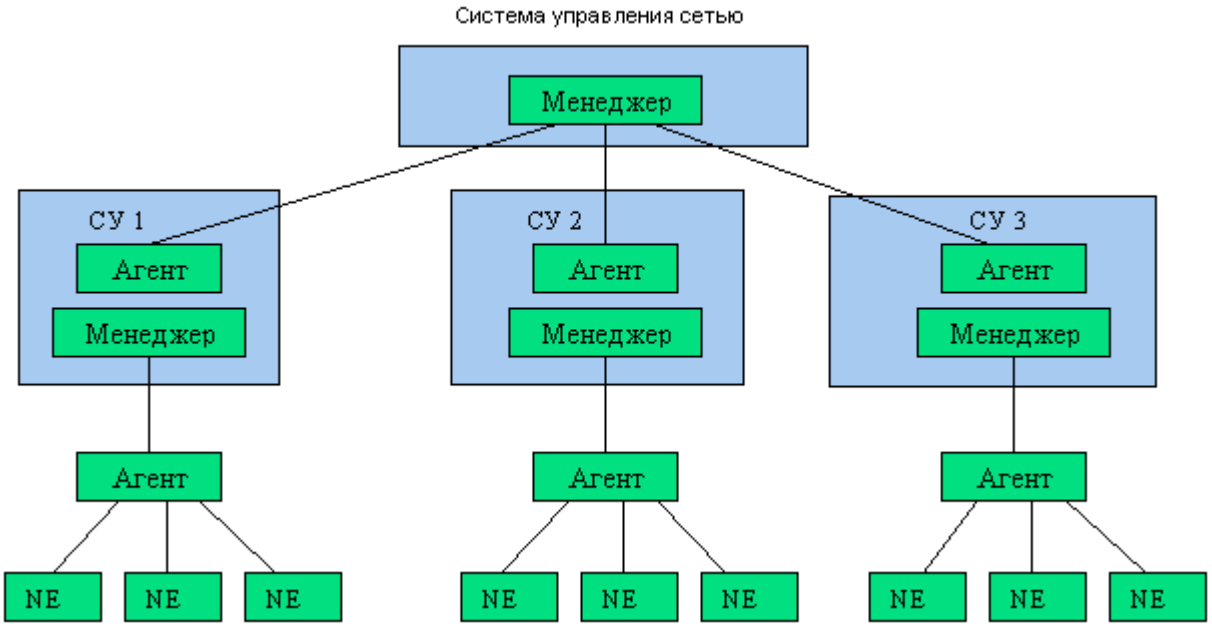

Рис. 2.4. Ієрархічні зв'язки між менеджерами

Функції, що виконуються системою автоматизації, моделюються за допомогою програмного додатку, який може міститися в одному пристрої, або розподілятися між кількома. (рис. 2.4.)

В одному пристрої може бути кілька ресурсів і кілька програмних додатків. Кожна програма може виконуватися на кількох пристроях і може займати частину ресурсів в одному пристрої. Посилання на МЕК 61499

Залежно від специфіки і складності об'єкта управління кількість одночасно запущених додатків і сервісів на кожному окремо взятому пристрої РСУ може варіюватися від одиниць до декількох сотень і реалізовувати такі функції:

- автоматизація управління технологічними процесами;
- взаємодія оператора з системою;

- моніторинг;
- забезпечення безпеки;
- дистанційне управління.

При цьому, найчастіше, ці процеси не є рівноцінними для підтримки працездатності системи, що призводить до необхідності їх поділу за пріоритетністю. Виходячи з пріоритету визначається ступінь доступності тих чи інших ресурсів системи для конкретно взятого сервісу. [9]

Наприклад, в системах, де основним показником якості роботи є швидкість реакції (відповіді) на запит, пріоритет системи моніторингу буде значно нижче пріоритету системи автоматизації. а також впровадження методів відстеження та дотримання цих пріоритетів. (рис. 2.5.)

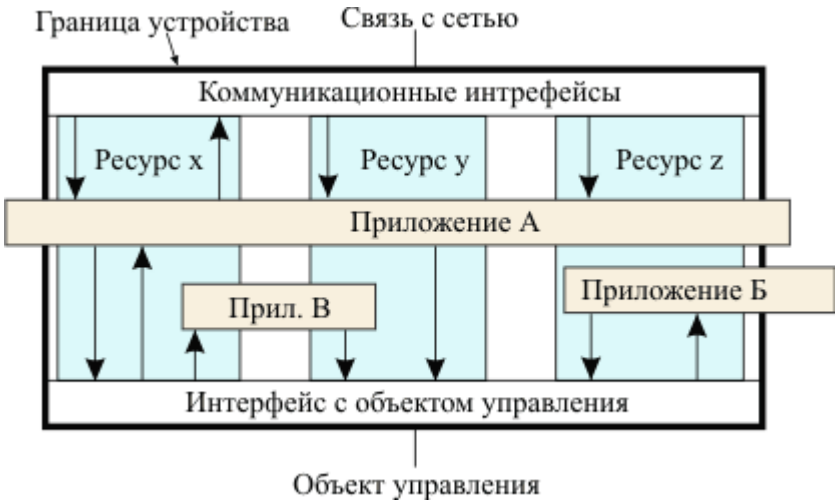

Рис. 2.5. Приклад моделі пристрою СА

2.2. Аналіз предмету проектування

Предметом розробки є система моніторингу стану розподіленої мережі шляхом систематизації даних щодо поточного стану кінцевих вузлів (хостів).

Моніторинг - це збір, обробка, агрегування і відображення кількісних даних в реальному часі про систему, таких як кількість запитів і типи запитів, кількість і типи помилок, час обробки, час життя і використання ресурсів сервера / клієнта / додатки / сервісу.

В системі застосовується моніторинг «чорної скриньки» - це моніторинг заснований на показниках, відображених внутрішніми компонентами системи, включаючи журнали, інтерфейси, такі як інтерфейс профілювання віртуальної машини.

Моніторинг та оповіщення дозволяють системі розповісти нам, коли вона зламана або може щось зламатися. Коли система не може автоматично сама себе фіксувати, потрібно, щоб людина досліджував попередження, визначав, чи є справжня проблема, пом'якшити її і визначити основну її причину.

Будь-яка несправність може мати кілька основних причин: наприклад, можливо, це було викликано одночасно недостатньою автоматизацією процесів, програмного забезпечення, яке розбилося на фіктивні вхідні дані, і недостатньою перевірки сценарію, використовуваного для створення конфігурації. Кожен з цих факторів може бути першопричиною, і кожен з них повинен бути відновлений.

Додаток надає зведений список основних показників служби. На панелі моніторингу в реальному часі відображаються кількісні дані про використання ресурсів сервісів на стороні клієнтів.

Система Повідомлення, призначена для читання людиною яка адмініструє систему, сповіщаючи про помилку або чергу відмов.

На одній машині може спостерігатися кілька сервісів. Сервіси можуть бути:

- Пов'язані один з одним: наприклад, сервер кешування і веб-сервер.
- Чи не пов'язані служби обміну обладнанням: наприклад, репозиторій коду і майстер для системи конфігурації, такий як Puppet або Chef.

Існує безліч причин для моніторингу системи, в тому числі: [8]

- Аналіз довгострокових тенденцій
- порівняння швидкості
- Попередження про відмови
- Побудова приладових панелей

- Проведення спеціального ретроспективного аналізу (т. Е. Налагодження)
- Моніторинг системи також корисний для надання нових даних для бізнес-аналітики і для полегшення аналізу порушень безпеки.

Розроблена система має слугувати інструментом для забезпечення стабільної роботи розподіленої мережі: [9]

1. Своєчасного виявлення відмов, перебоїв та суттєвих відхилень показників продуктивності мережі.
2. Прогнозування нестабільності в роботі мережі на основі даних про стан обчислювальних та мережевих ресурсів мережі.
3. Створення звітів та зручної інфографіки щодо поточного стану системи
4. Створення бази даних щодо показників роботи мережі, яку можна використовувати для подальшого аналізу.

Система моніторингу має відповідати наступним вимогам:

1. прийнятна швидкість роботи;
2. незначне додаткове навантаження на ресурси системи (в тому числі додатковий службовий трафік);
3. інтуїтивний інтерфейс (легкість використання незалежно від кваліфікації оператора);
4. портативність;
5. масштабованість;
6. наявність можливості конфігурації.

2.3. Обґрунтований вибір параметрів моніторингу

При моніторингу для дослідження питань, пов'язаних з продуктивністю, початкові зусилля повинні бути зосереджені на трьох основних областях.

• Активність роботи з диском

• Використання процесора

• Використання пам'яті

Контроль комп'ютера, в якому виконується системний монітор, може злегка впливати на продуктивність комп'ютера. Тому або реєструйте дані системного монітора на іншому диску (або комп'ютері), щоб зменшити вплив процедури контролю на контрольований комп'ютер, або виконуйте системний монітор з віддаленого комп'ютера. Контролюйте тільки ті лічильники, які необхідні. При моніторингу занадто великого числа лічильників в процесі моніторингу виникають додаткові витрати ресурсів, що впливає на продуктивність того комп'ютера, за яким здійснюється спостереження.

Метрики додатків - це показники, що стосуються одиниць обробки або роботи, які залежать від ресурсів рівня хоста, як-от служби чи програми. Конкретні типи показників, які слід переглянути, залежать від того, що надає послуга, якими залежностями вона володіє та якими іншими компонентами вона взаємодіє. Показники на цьому рівні - це показники стану здоров'я, продуктивності чи завантаженості програми:

• Похибки та успішність

• Збої в роботі та перезавантаження

• Продуктивність та затримка відповідей

• Використання ресурсів

Ці показники допомагають визначити, чи працює програма чи ефективно та ефективно.[4]

2.3.1. Метричні показники мережі та зв’язку

Для більшості типів інфраструктури, мережевих показників та індикаторів підключення буде ще один набір даних, який варто вивчити. Це важливі показники доступності, орієнтованої назовні, але також мають важливе значення для забезпечення доступу сервісів до інших машин для будь-яких систем, що охоплюють більше ніж одну машину. Як і інші показники, про які ми обговорювали

дотепер, мережі слід перевірити на загальну функціональну коректність та здатність забезпечити необхідну продуктивність, переглянувши:

• Підключення

• Коефіцієнт помилок і втрата пакету

• Затримка

• Використання пропускної здатності

Контроль рівня мереж може допомогти вам покращити доступність та чуйність як ваших внутрішніх, так і зовнішніх служб.[15]

2.3.2. Визначення рівня завантаження ЦП

Постійний високий рівень використання ЦП може вказувати на необхідність оновлення ЦП або на необхідність додавання декількох процесорів. Крім того, високий рівень використання ЦП може вказувати на погано налаштоване або погано розроблене додаток. Оптимізація роботи програми може знизити рівень завантаження ЦП.

Ефективним способом визначення рівня завантаження ЦП є використання лічильника Процесор:% завантаженості процесора в службовій програмі «Системний монітор». Цей лічильник відстежує час, яке ЦП витрачає на виконання потоку під час роботи. Постійний рівень завантаження ЦП в діапазоні від 80 до 90% може вказувати на необхідність оновлення ЦП або на необхідність додавання декількох процесорів. При роботі з багатопроцесорними системами стежте за окремим екземпляром згаданого лічильника для кожного процесора. Це значення сумарне процесорний час зазначеного процесора. Щоб визначити середнє для всіх процесорів, скористайтеся натомість лічильником Система:% загального процесорного часу.

Додатково можна контролювати такі лічильники:

Процесор:% роботи в привілейованому режимі

Відповідає відсотку процесорного часу, витраченого на виконання команд ядра операційної системи Microsoft Windows, таких як обробка запитів вводу-виводу SQL Server. Якщо значення цього лічильника постійно високу, в той час як

лічильники для об'єкта Фізичний диск також мають високі значення, то необхідно розглянути питання про встановлення більш швидкого і більш ефективної дискової підсистеми.

Процесор:% роботи в режимі користувача

Відповідає відсотку часу роботи процесора, яке він витрачає на виконання додатків, наприклад SQL Server.

Система: Довжина черги процесора.

Відповідає кількості потоків, які очікують обробки процесором. Якщо потокам деякого процесу потрібно більше циклів процесора, ніж це можливо, значить, вузьким місцем системи є процесор. Якщо кількість процесів, що вимагають обробки процесором, велике, необхідно встановити більш швидкий процесор. Або, в багатопроцесорної системі, необхідно додати ще один процесор. [15]

2.3.3. Визначення поточного розподілення фізичної пам’яті

Фізична пам'ять поділяється на такі категорії:

Категорія пам'яті	Деталі
Hardware Reserved	Зарезервовано для апаратних засобів (наприклад, відео, адаптерів Ethernet) та недоступних для користувачів процесів.
In Use	Загальна оперативна пам’ять, що використовується в робочих процесах
Modified	Розділи пам'яті, які були змінені та потребують запису на диск
Standby	Розділи, до яких недавно не можна отримати доступ та звільнені з робочого набору. Вони можуть бути завантажені на диск, якщо потрібен простір, але вони запам'ятовуються в пам'яті, якщо їх процес потребує їх знову.

Free	ОЗУ, яка не використовується

Табл. 2.1. Категроії поділу фізичної пам'яті

Групи пам'яті з точки зору доступності для нових процесів:

Категорія пам'яті	Складається з	Опис
Available	Standby + Free	Доступна оперативна пам'ять для запуску нових процесів
Cached	Modified + Standby	Оперативна пам’ять яка зараз використовується як кеш, який при необхідності може бути звільнений
Total	In Use + Modified + Standby + Free	Загальна оперативна пам’ять, доступна в системі для користувацьких процесів
Installed	Total + Hardware Reserved	Оперативна пам’ять встановлена в системі

Табл. 2.2. Групи поділу фізичної пам'яті

Commit Limit

Це загальний об'єм пам'яті, який можна використовувати в системі, і це сума оперативної пам'яті та простору файлу сторінок. Якщо файли сторінок встановлені для автоматичного розширення, і для них є дисковий простір, то межа обмеження може збільшитися.

Commit Size

Кожен процес має розмір фіксації, який є сумою пам'яті, яка використовується для цього процесу (фізична пам'ять + файл підкачки) та додаткова пам'ять, яка зарезервована менеджером пам'яті для цього процесу. Поки зарезервована пам’ять

наразі не використовується, менеджер пам'яті гарантує, що вона залишається доступною. Розмір фіксації для процесу може змінюватися протягом життя процесу.

Committed Bytes

Це загальний обсяг фіксації пам'яті для всіх процесів у системі. Як і у випадку розміру фіксації процесу, вкладені байти включають пам'ять, яка ще не виділена процесу, але зарезервована для подальшого використання. Якщо допущені байти перевищують ліміт фіксації в системі, система по можливості розширить простір файлів сторінок або запобіжить запуску додаткових процесів, якщо ні. Якщо ви не можете розширити файл підкачки, переконайтесь, що:

Committed Bytes < Commit Limit

Working Set

Working Set - це набір сторінок пам'яті, які має процес в оперативній пам'яті. Менеджер пам'яті Windows відстежує робочий набір і переміщує сторінки в режим очікування, якщо вони активно не використовуються. Зауважте, що робочий набір може містити пам'ять для спільних файлів (наприклад, системний dll), до яких можна отримати безліч процесів. Ця спільна пам'ять вважається частиною робочого набору для кожного процесу.

Private Working Set

Це робочий набір для процесу, що не включає спільну пам'ять. Якщо процес закінчиться, це буде пам'ять, яка буде звільнена.

Page Faults

Це кількість разів за секунду, яку процес повинен звернутися до пямяті для свого **Working Set**. Існує два типи помилок сторінки:

Слабкі несправності:

За слабкої помилки, сторінка пам'яті, якій потрібен процес, вже знаходиться в оперативній пам'яті - наприклад, вона може бути в робочому наборі для іншого процесу. Менеджер пам'яті додає існуючу сторінку пам'яті до робочого набору без необхідності дискового вводу. М’які несправності не впливають на продуктивність системи.

Сильні несправності:

Сильна помилка виникає, коли менеджеру пам’яті потрібно зчитувати дані з диска, або зі сторінки сторінки, або читати файл безпосередньо. Сильні несправності можуть бути симптомом дефіциту пам’яті: якщо недостатньо оперативної пам’яті для процесів, що працюють в системі, менеджер пам’яті може підписувати робочий набір пам’яті для одного процесу, щоб звільнити місце для робочого набору іншого процесу.

Для помилок сторінки лічильники не розрізняють м'які та жорсткі несправності, і слід очікувати деякого рівня жорстких несправностей, тому розробіть базову лінію та використовуйте її як поріг. Якщо у несправностях сторінки з’являється шип, співвіднесіть його з IO диска (наприклад, Логічний диск \ Поточна довжина черги диска, Логічний диск \ Читання дисків / сек, Логічний диск \ Запис диска / сек).

Paged Pool/Nonpaged Pool

Paged Pool - це пул пам’яті, який може бути завантажений на диск, тоді як Nonpaged Pool - це сторінки пам’яті, які не можуть бути завантажені на диск і повинні залишатися в робочому наборі. За цими лічильниками слід відстежувати базовий рівень та використовувати їх як індикатори процесів із витоком пам'яті. [11]

2.4. Визначення вимог щодо моніторингу розподілених систем

Специфіка сучасних мереж передбачає, що на стороні клієнта на постійній основі діє чимала кількість додатків і сервісів, які активно споживають як апаратні ресурси, так і ресурси мережі. При цьому в багатьох структурах майже не враховується реальна пріоритетність процесів, що може привести до збоїв в роботі ключових елементів системи, в момент, коли якомусь з додатків або сервісу знадобляться ресурси, зайняті фоновим мало-пріоритетним процесом. Це призводить до необхідності детального моніторингу ресурсо-споживання як в цілому, так і диференційовано по кожному з сервісів, запущеного з боку клієнта.

Оскільки кількість сервісів і додатків, запущених на одному клієнті може аж ніяк не обмежуватися десятками або навіть сотнями, під час моніторингу такої кількості об'єктів стоїть завдання ретельної і продуманої класифікації та організації отриманих даних, як в процесі безпосереднього логування, так і при подальшому зберіганні та аналізі отриманої інформації.

При роботі з системою моніторингу в операторів може виникати потреба в більш детальному аналізі поведінки того чи іншого параметра, або ж, навпаки - деякі параметри можуть бути повністю виключені з деяких тестів або звітів для підвищення продуктивності системи.

Виходячи з визначених вимог до систем моніторингу, було прийнято рішення реалізувати наступний функціонал:

- Моніторинг стану хостів: (завантаження процесора, використання диска, системні логи).

Отримання деталізованої інформації по миттєвому (або за певний період) споживання таких ресурсів, як: завантаженість процесора, оперативної пам'яті, відеопам'яті, споживання зовнішньої пам'яті, трафік. Інформація про дані о параметрах може бути запрошена пакетно, або по кожному параметру окремо.

- Моніторинг мережевих служб: (SMTP, POP3, HTTP, NNTP, ICMP, SNMP)
- Логування, створення звітів

Інструмент створення звітів за певний період часу а також отримання миттєвої статистики в режимі реального часу по кожному з клієнтів / сервісів.

- Можливості для масштабування: Проста архітектура модулів розширень (плагінів) для полегшання процесу розробки додаткових способів перевірки служб;
- Можливість визначати обробники подій, що відбулися зі службами або хостами для проактивного вирішення проблем.

2.5. Метод моніторингу розподіленої мережної системи

Процес моніторингу передбачає передачу даних про стан окремих елементів об'єкта моніторингу на віддалений сервер, який відповідає за первинну обробку

вхідних даних, складання звітності та її організоване зберігання. Даний процес можна розбити на кілька логічних етапів: ініціалізацію, пост ініціалізацію (переконфігуруванні системи), безпосередньо роботу системи і завершення роботи системи. З огляду на те, що в реальних системах завжди існує ризик не прогнозованого відмови, останній етап (завершення роботи) повинен передбачати як коректне, так і некоректне відключення системи або її компонентів.

У загальному вигляді, робота розробленої системи описується наступними інструкціями:

1. Ініціалізація системи
   Ініціалізація серверу:
   1.1. Зчитується файл конфігурації сервера (додаток А - приклад файлу конфігурації сервера);
   1.2. Встановлюються основні параметри сервера - частота опитування клієнта, формат запитуваної звіту, шляхи зберігання звітів, рівень прав доступу тощо;
   1.3. Зчитується база даних про діючі (довірених) клієнтів;
   1.4. Завершальним етапом ініціалізації бути відкриття порту на прослуховування вхідних з'єднань.
   Ініціалізація клієнта:
   1.1. Зчитується файл конфігурації клієнта (додаток Б - приклад файлу конфігурації клієнта);
   1.2. Встановлюються основні параметри клієнта - мережеву адресу сервера, час зберігання звітів, шляхом зберігання звітів, рівень прав доступу тощо;
   1.3. Клієнт починає локальний моніторинг в офлайн режимі (якщо зазначено в конфігурації клієнта);
   1.4. Завершальним етапом ініціалізації бути відкриття порту на прослуховування вхідних з'єднань від сервера.
2. Пост-ініціалізація, синхронізація клієнта і сервера
   2.1. Сервер шле Hello-message кожному довіреному клієнтові, в якому вказуються параметри подальшого моніторингу - перелік ресурсів, що підлягають моніторингу для кожного сервісу, частота звернення до сервера, частота створення «знімків» стану системи для локального зберігання;
   2.2. Клієнт оновлює налаштування відповідно до отриманих установками від сервера;
   2.3. Клієнт відповідає сервера про успішне з'єднання;

2.4. Сервер шле повідомлення про старт моніторингу;
2.5. Клієнт починає в фоновому режимі слати звітність на сервер без підтвердження про доставку (за замовчуванням);

3. Робота системи

Система моніторингу може працювати в двох режимах - отримання звітів реального часу або запиту статистики з клієнтів за період часу.

Real-time звіти

3.1. Клієнт із заданою періодичністю створює кадр стану пристрою (додаток В - приклад кадру стану пристрою)
3.2. Клієнт записує кадр в буфер кадрів і в локальну БД
3.3. З заданої періодичністю, останній кадр з буфера пересилається на сервер
3.4. Сервер обробляє повідомлення про клієнта, формуючи кадр стану клієнта і записує отримані дані в БД. Такі дані можуть бути виведені на пристрої виведення у вигляді графіків в режимі реального часу.

Звіти за період:

Шляхом запиту з БД сервера*:*

3.1. З БД сервера запитуватися необхідні поля
3.2. Заданий шаблон звіту заповнюється даними з БД

Шляхом запиту з БД окремого пристрою:

3.1. Сервер шле запит на клієнт, із зазначенням необхідних полів і періоду запиту
3.2. Клієнт запитує з локальної БД необхідні поля
3.3. Клієнт формує відповідь сервера у вигляді необхідного набору даних
3.4. Отримавши відповідь від клієнта, сервер заповнює заданий шаблон звіту отриманими даними

4. Припинення роботи системи або її компонентів

Коректне відключення клієнта*:*

4.1. Клієнт повідомляє серверу про припинення роботи
4.2. Клієнт зупиняє моніторинг
4.3. Клієнт завершує збереження даних
4.4. Клієнт завершує роботу

Некоректне відключення клієнта:

4.1. Якщо сервер не отримає відповіді від певного клієнта відповіді у визначений конфігурацією проміжок часу - буде сформовано повідомлення про помилку на даному клієнті.

Коректне відключення / перезавантаження сервера:

4.1. Сервер шле всім активним клієнтам повідомлення про припинення роботи. Повідомлення так само вказує клієнтам, чи слід перервати локальний моніторинг (Повідомлення від сервера є має більший пріоритет, ніж локальні настройки клієнта).

4.2. Сервер очікує відповіді від клієнтів протягом часу, заданого в конфігурації;

4.3. Клієнт, отримавши повідомлення про припинення роботи припиняє моніторинг (якщо передбачено конфігурацією) шле відповідь про відключення і припиняє комунікацію з сервером.

4.4. Пункти 4.1-4.2 повторюються для клієнтів, які не були відключені до того часу, поки не залишиться активних клієнтів, або доки не вичерпається задана конфігурацією кількість спроб з'єднання.

Некоректне відключення сервера:

4.1. Втративши з'єднання з сервером, клієнт формує повідомлення про помилку, записує його в локальний лог; призупиняє процеси обміну мережевими повідомленнями з сервером і чекає перепідключення сервера.

Висновки до розділу 2

Процес моніторингу передбачає передачу даних про стан окремих елементів об'єкта моніторингу на віддалений сервер, який відповідає за первинну обробку вхідних даних, складання звітності та її організоване зберігання.

Виходячи з визначених вимог до систем моніторингу, було висунуто наступні вимоги:

- Моніторинг стану хостів: (завантаження процесора, використання диска, системні логи).

Отримання деталізованої інформації по миттєвому (або за певний період) споживання таких ресурсів, як: завантаженість процесора, оперативної пам'яті, відеопам'яті, споживання зовнішньої пам'яті, трафік. Інформація про дані о параметрах може бути запрошена пакетно, або по кожному параметру окремо.

- Моніторинг мережевих служб: (SMTP, POP3, HTTP, NNTP, ICMP, SNMP)
- Наявність можливості логування, створення звітів
- Наявність можливості для масштабування: Проста архітектура модулів розширень (плагінів) для полегшання процесу розробки додаткових способів перевірки служб;
- Наявність можливості визначати обробники подій, що відбулися зі службами або хостами для проактивного вирішення проблем.

Виходячи із виявлених вимог до систем моніторингу запропоновано метод моніторингу диференційованого вживання ресурсів окремими сервісами у розподілених системах.

За запропонованим методом процес моніторингу можна розбити на кілька логічних етапів: ініціалізацію, пост ініціалізацію (переконфігуруванні системи), безпосередньо роботу системи і завершення роботи системи. З огляду на те, що в реальних системах завжди існує ризик не прогнозованого відмови, останній етап (завершення роботи) повинен передбачати як коректне, так і некоректне відключення системи або її компонентів.

РОЗДІЛ 3.
РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ «NETSERVICES EASY MONITOR»

3.1. Інструменти та технології

3.1.1. Вибір інтегрованого середовища розробки

Visual Studio - лінійка продуктів, розроблена компанією Microsoft яка включає програмне забезпечення і ряд інших інструментальних засобів для полегшення процесу розробки. Дані продукти дозволяють розробляти різноманітні додатки, в тому числі консольні додатки, додатки з графічним інтерфейсом Windows Forms, а також веб-сайти, веб-додатки, веб-служби для Windows, Windows Mobile, Windows CE,, .NET FrameworkXbox, Windows Phone і .NET Compact FrameworkSilverlight.[15]

Visual Studio включає в себе підтримку технології редактору вихідного коду IntelliSense з можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Решта вбудованого інструментарію:

- дизайнер класів;
- редактор форм для спрощення створення графічного інтерфейсу додатку;
- веб-редактор;
- дизайнер схем баз даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні. Включаючи розширення підтримки систем (як, наприклад, контролю версій вихідного коду), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду) або інструментів для інших аспектів (наприклад, клієнт Team Explorer для роботи з Visual SourceSafe з предметно-орієнтованими мовами програмної системи в галузі

програмного забезпечення Team Foundation Server).

3.1.2. Вибір мови програмування

В ході розробки використовувалась мова програмування C#.

Фактично, C# являє собою гібрид різних мов програмування, від кожної з яких вона взяла найкраще.

Будучи спадкоємицею мови C++, мова C# використовує подібний до неї синтаксис, проте позбавлена неоднозначностей, які допускали компілятори C++.

На відміну від Java, C# генерує код, що може бути використаний лише у середовищі виконання .NET – так званий керований код (managed code). Зокрема, це передбачає автоматичне керування пам’яттю та відсутність потреби працювати напряму із вказівниками на адреси у пам’яті, що дуже зменшує кількість ймовірних помилок при розробці та використанні програми.

Компонент .NET Microsoft Visual Studio – це інтегроване середовище розробки програмного забезпечення, яке забезпечує абсолютно комфортний інтерфейс для створення C#-програм.

C# створювалася як мова компонентного програмування, і в цьому одне з головних переваг мови, спрямована на можливість повторного використання створених компонентів. З інших об'єктивних факторів відзначимо наступні:

- C# створювався паралельно з каркасом Framework .Net і повною мірою враховує всі його можливості - як FCL, так й CLR;
- C# є повністю об'єктно-орієнтованою мовою, де навіть типи, вбудовані в мову, представлені класами;
- C# є потужною об'єктною мовою з можливостями спадкування й універсалізації;
- C# є спадкоємцем мов C/C++, зберігаючи кращі риси цих популярних мов програмування;
- завдяки каркасу Framework .Net, що стали надбудовою над операційною системою, програмісти C# одержують ті ж переваги роботи з віртуальною машиною, що й програмісти Java. Ефективність коду навіть підвищується, оскільки

виконавче середовище CLR являє собою компілятор проміжної мови, у той час як віртуальна Java-машина є інтерпретатором байта-коду;

- потужна бібліотека каркасів підтримує зручність побудови різних типів додатків на C#, дозволяючи легко будувати Web-служби, інші види компонентів, досить просто зберігати й одержувати інформацію з бази даних й інших сховищ даних. [12]

3.1.3. Вибір основних програмних бібліотек

Бібліотека Pcap

Бібліотека Pcap (Packet Capture) дозволяє створювати програми для аналізу даних мережі, що надходять на мережеву карту ЕОМ. Прикладом ПЗ, що використовує бібліотеку Pcap, може слугувати програма Wireshark. Цю бібліотеку використовують різноманітні сніфери, програми тестування та моніторингу мережі. Вона призначена для використання в рішеннях, реалізованих мовами сімейства C (C/C ++ та C#). У випадках роботи з бібліотекою на інших мовах, таких як Java, використовують спеціальні обгортки. Для Unix-подібних систем – такою обгорткою є бібліотека libpcap, а для Microsoft Windows - WinPcap.

Програмні системи мережевого моніторингу можуть використовувати libpcap або WinPcap для перехоплення пакетів, які пересилаються по мережі, і також (в останніх версіях) для, безпосередньо, передачі пакетів в мережі. Libpcap і WinPcap також підтримують читання і збереження файлів захоплених пакетів.

Програми, написані на основі libpcap або WinPcap, можуть перехоплювати мережевий трафік та аналізувати його. Файл захопленого трафіку зберігається в форматі, зрозумілому для додатків, що використовують Pcap.

Для перехоплення пакетів мережева карта повинна працювати в так званому режимі promiscuous-mode. Для роботи в такому режимі потрібна підтримка на рівні драйверів, яку забезпечує спеціалізована бібліотека WinPcap

Склад WinPcap:

- фільтр пакетів на рівні ядра
- низько-рівнева DLL (packet.dll);

- високо-рівнева системно-незалежна бібліотека (wpcap.dll).

Недоліком даної бібліотеки є те, що вона працює не з усіма нестандартними адаптерами.

Бібліотеки SharpPcap и Packet.Net

SharpPcap - бібліотека для .NET, яка дозволяє перехоплювати мережеві пакети. Загалом, це «обгортка» над бібліотекою Pcap.

Із SharpPcap також поставляється бібліотека для парсингу пакетів - Packet.Net. Packet.Net підтримує такі протоколи: Ethernet, LinuxSLL, Ip (IPv4 and IPv6), Tcp, Udp, ARP, ICMPv4 и ICMPv6, IGMPv2, PPPoE, PTP, Link Layer Discovery Protocol (LLDP), Wake-On-LAN (WOL). [11]

3.1.4. Вибір технології створення графічного інтерфейсу користувача

Технологія WPF (Windows Presentation Foundation) є частиною екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

Якщо при створенні традиційних додатків на основі WinForms за отрисовку елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI +, то додатки WPF засновані на DirectX. У цьому полягає ключова особливість рендеринга графіки в WPF: використовуючи WPF, значна частина роботи по відображенні графіки, як найпростіших кнопочок, так і складних 3D-моделей, лягати на графічний процесор на відеокарті, що також дозволяє скористатися апаратним прискоренням графіки.

Однією з важливих особливостей є використання мови декларативною розмітки інтерфейсу XAML, заснованого на XML: ви можете створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.[12]

Переваги WPF:

• Використання традиційних мов .NET-платформи - C # і VB.NET для створення логіки додатка

• Можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованому на xml і представляє

альтернативу програмному створення графіки та елементів управління, а також можливість комбінувати XAML і C # / VB.NET

• Незалежність від дозволу екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, додатки на WPF легко масштабуються під різні екрани з різним дозволом.

• Нові можливості, в порівнянні з WinForms, наприклад, створення тривимірних моделей, прив'язка даних ітд.

• Гарне взаємодія з WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління з WinForms.

• Багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми і багато іншого

• Апаратне прискорення графіки

• Створення додатків під безліч ОС сімейства Windows - від Windows XP до Windows 10 [15]

Однак варто враховувати, що в порівнянні з додатками на Windows Forms обсяг програм на WPF і споживання ними пам'яті в процесі роботи в середньому трохи вище.

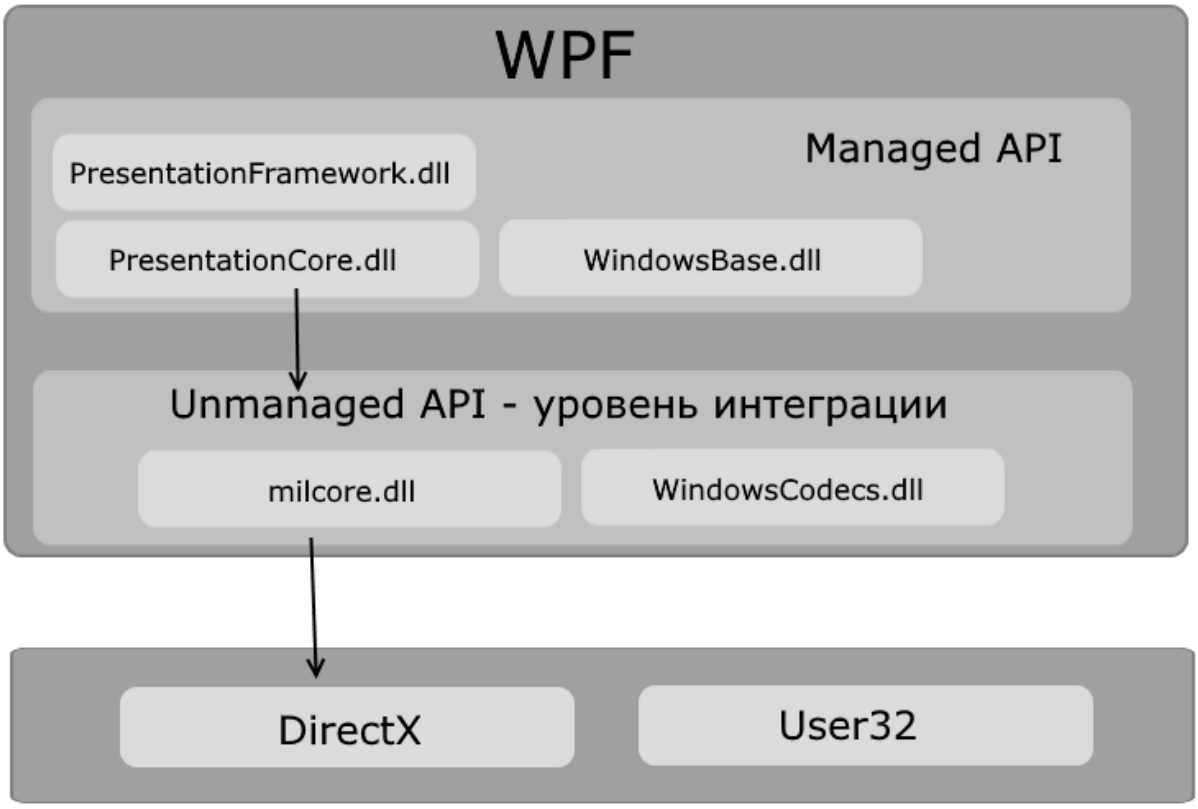

Рис. 3.1. Архітектура технології WPF

Як видно на схемі (рис. 3.1.,) WPF розбивається на два рівня: managed API і unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) містить код, що виконується під управлінням загальномовного середовища виконання .NET - Common Language Runtime. Цей API описує основний функціонал платформи WPF і складається з наступних компонентів:

• PresentationFramework.dll: містить всі основні реалізації компонентів і елементів управління, які можна використовувати при побудові графічного інтерфейсу

• PresentationCore.dll: містить всі базові типи для більшості класів з PresentationFramework.dll

• WindowsBase.dll: містить ряд допоміжних класів, які застосовуються в WPF, але можуть також використовуватися і поза даної платформи

Unmanaged API використовується для інтеграції вищого рівня з DirectX:

• milcore.dll: забезпечує інтеграцію компонентів WPF з DirectX.

• WindowsCodecs.dll: бібліотека, яка надає низкоуровневую підтримку для зображень в WPF

Ще нижче власне знаходяться компоненти операційної системи і DirectX, які здійснюють візуалізацію компонентів програми, або виконують іншу низькорівневу обробку.

3.1.5. Вибір методу мережної комунікації

Вся мережа складається з окремих елементів - хостів, які представляють собою ЕОМ та інші підключені пристрої. Між собою хости з'єднані каналами зв'язку (кабелі Ethernet, Wi-Fi тощо) і маршрутизаторами. Маршрутизатори об'єднують хости в підмережі і контролюють передачу даних між ними.

Протокол – це угода про те, як пакети даних передаватимуться по каналах комунікації. Таким чином, протокол впорядковує мережеву взаємодію.[7]

Існує безліч різних протоколів. Протоколи, які використовуються для передачі даних по мережі, складають сімейство протоколів TCP/IP. Основні з них: Internet Protocol (IP), Transmission Control Protocol (TCP) і User Datagram Protocol (UDP). Причому ці протоколи організовані в рівневу систему:

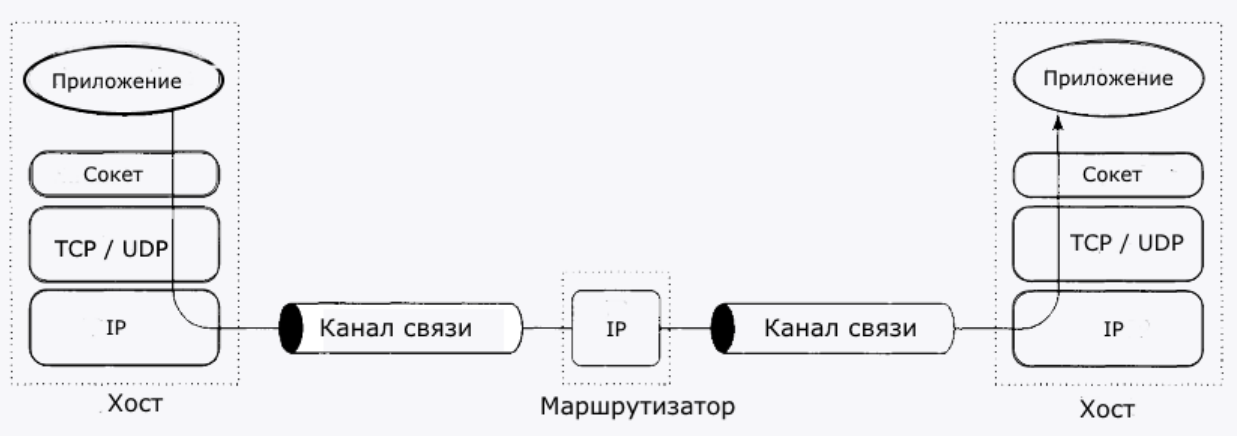

Рис. 3.2. Схема TCP/IP комунікації

Передача даних по протоколах TCP / IP

IP представляє мережевий рівень. Він використовує попередні рівні, які представляють фізичні канали комунікації для передачі пакетів з даними іншого хосту.

Вище IP розташовується транспортний рівень, який утворюють протоколи TCP і UDP. Ці протоколи використовують певні порти для передачі даних. TCP дозволяє відстежити втрату пакетів і їх дублювання при передачі. UDP подібного не дозволяє зробити і націлений на просту передачу даних.

Однак додаток взаємодіє з рівнем TCP/UDP не безпосередньо, а через спеціальний API, який надають сокети. (рис. 3.2.)

Сокети - це не є протоколом, це інтерфейс для створення мережевих додатків, який спирається на вбудовані можливості операційної системи.

Залежно від використовуваного протоколу розрізняють два види сокетів: потокові сокети (stream socket) і дейтаграммний сокети (datagram socket). Потокові сокети використовують протокол TCP, дейтаграммний - протокол UDP.

Коли додаток посилає повідомлення додатку, запущеному на іншому хості, то додаток звертається до сокета для передачі даних на рівень TCP/UDP. Далі з цього транспортного рівня дані передаються мережному рівню - рівню протоколу IP. IP-протокол передає дані далі фізичним рівням для передачі по мережі.

Щоб унікально визначати хости в мережі кожен хост має адресу. Існує кілька різних протоколів адрес. В даний час найбільш поширений протокол IPv4, який передбачає подання адреси у вигляді 32-бітного числа. Така адреса містить чотири числа, між якими ставлять крапку, і кожне число знаходиться в діапазоні від 0 до 255. Однак також останнім часом набирає обертів використання адрес протоколу IPv6, які представляють собою 128-бітове значення.

Крім адреси при мережевих взаємодіях використовуються порти. Порт представляє 16-бітове число в діапазоні від 1 до 65 535. Використання портів дозволяє розмежувати кілька запущених додатків на одному хості.

Платформа .NET і мова програмування C # надають всі необхідні можливості для створення додатків, які можуть взаємодіяти в мережі і використовувати різні мережеві протоколи.[10]

3.2. Етапи розробки

Процес розробки системи моніторингу структурно можна поділити на три основних етапи:

1. аналітична підготовка;
2. програмування та тестування;
3. документування.

Аналітична підготовка – вкрай важливий етап розробки. Від його результатів зазвичай залежить подальший результат всієї роботи та об’єм витраченого на неї часу.

Для розробки системи моніторингу цей етап поділяється на такі кроки:

1. Аналіз предмету проектування
2. Складання вичерпного переліку доступного функціоналу
3. Розробка методу моніторингу
4. Розробка структури програми
5. Визначення інструментів та технологій необхідних при розробці

Метою цього етапу є отримання готової декомпозованої формалізованої моделі, яку легко описати у вигляді алгоритму та, безпосередньо реалізувати в якості програмного забезпечення.

Другий етап уособлює в собі безпосередньо процес розробки, а саме:

1. Написання коду
2. Відладка
3. Тестування
4. Конфігурування

Результатом завершення цього етапу має бути готовий програмний модуль, готовий до використання. Під час другого етапу можливе тимчасове повернення до аналітичного етапу задля вирішення архітектурних проблем, виявлених, нажаль, лише при написанні коду або тестуванні. Наостанок залишається лише завершаючий етап – документування (рис. 3.3.)

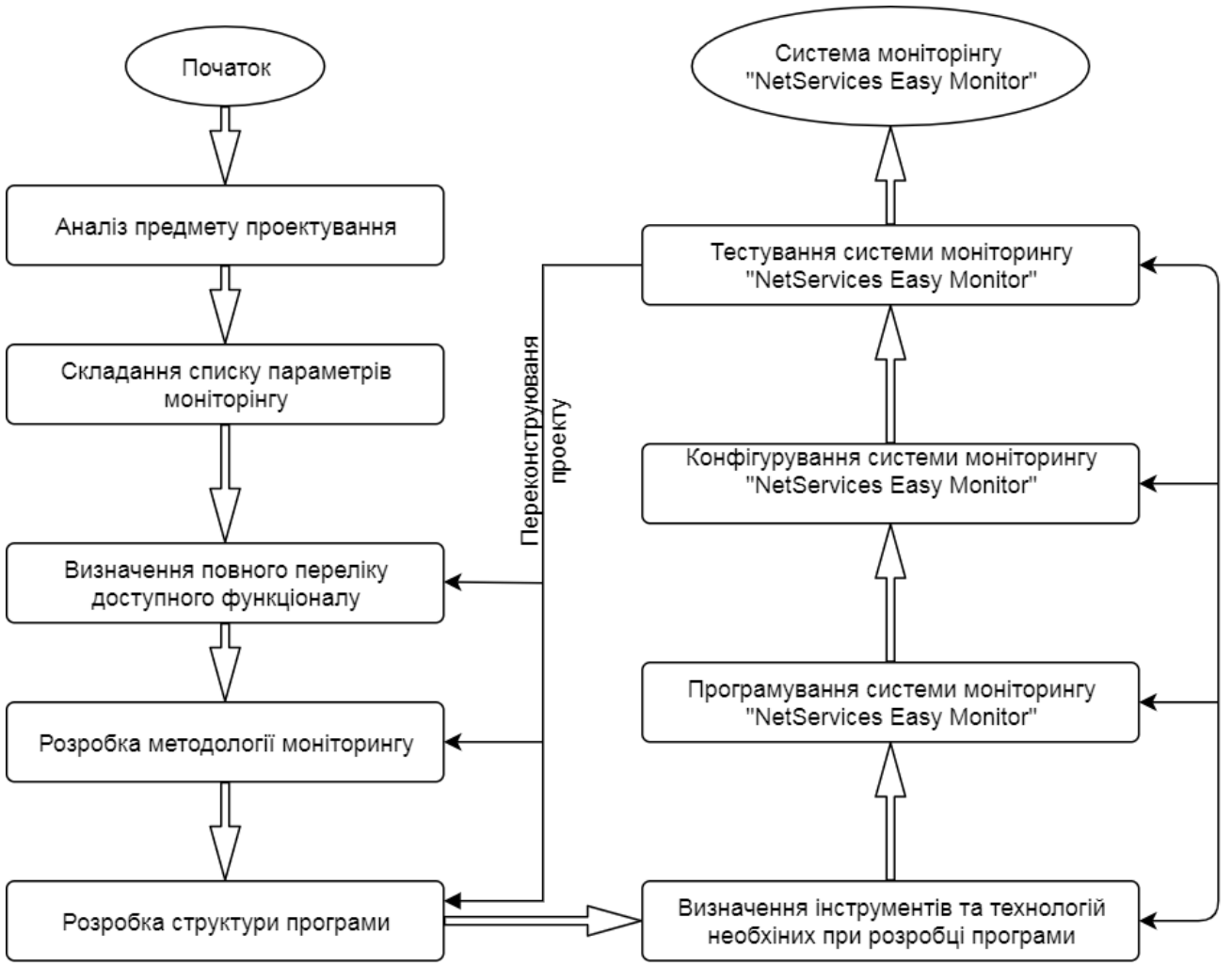

Рис. 3.3. Етапи розробки

3.3. Розробка структури програми

Створювана система моніторингу передбачає клієнт-серверну архітектуру.

На сервер покладена функція визначення топології, складу мережі, менеджмент клієнтів (клієнтської частини системи, що розробляється), а так само зберігання, обробка даних, що надходять з клієнтів, встановлених на фізичні або віртуальні хости-компоненти системи, що підлягає моніторингу.

Клієнтські програми покликані перехоплювати спостерігаються параметри, створювати локальну статистику за період часу і надавати звітність з додатком-сервера в режимі реального часу або за Get-запитам з сервера; а так само є інтерфейсом для передачі керуючих сигналів до сервісів.

Операції, що виконуються при роботі системи моніторингу мережі можна розділити на чотири окремих категорії:

- моніторинг мережевих служб;
- моніторинг стану хостів;
- логування
- вивід інфографіки

Кожен з вищезазначених процесів є, по суті, самостійною частиною системи моніторингу мережі, має свої особливості, методи та реалізації. Отже, в реалізації системи моніторингу доцільно виділити:

1. три основних класи, що містять поля, методи та властивості для обробки даних моніторингу:
- клас, що реалізує роботу з даними моніторингу мережевих служб
- клас, що реалізує роботу з даними моніторингу стану хостів
- клас що реалізує мережеву комунікацію між клієнтом та сервером системи моніторингу
2. класи, які описують структуру запитів моніторингу та відповідає за перехоплення параметрів які підлягають моніторингу:
- клас, що описує структуру запиту стану віддаленого хосту (завантаження процесора, використання диска, оперативної пам’яті, мережевих ресурсв) та містить методи для перехоплення вищезазначених параметрів;

3. класи, які відповідають за створення звітності:
- клас, який реалізує поточне логування моніторингу, обробку та систематизацію отриманих логів
- клас, який реалізує візуалізацію отриманих результатів моніторингу
4. класи, які реалізують API для підключення додаткових плагінів.

3.4. Програмування програмного модулю "NetServices Easy Monitor"

3.4.1. Організація мережевої комунікації

Мережева комунікація – одна з найголовніших компонент програми. Для "NetServices Easy Monitor" вона має стандартну реалізацію на TCP сокетах (рис. 3.).

Рис. 3.4. Загальна схема роботи серверного сокету TCP

Серверний сокет слухає з'єднання на порту, та очікує з'єднання із клієнтом.(рис. 3.5)

```
try
{
    tcpListener = new TcpListener(IPAddress.Any, 8888);
    tcpListener.Start();
    Console.WriteLine("Server started");
    while(true)
    {
        TcpClient tcpClient = tcpListener.AcceptTcpClient();
        Client client = new Client(tcpClient, this);
        Thread clientThread = new Thread(client.Process);
        clientThread.Start();
    }
}
```

Рис. 3.5.Логіка конекту клієнту до серверу (з боку серверу)

Рис. 3.6. Загальна схема роботи клієнтського сокету TCP

```
client = new TcpClient();
try
{
    client.Connect(host, port);
    networkStream = client.GetStream();

    byte[] data = Encoding.Unicode.GetBytes(hostId + hostName);
    networkStream.Write(data, 0, data.Length);

    receiveThread = new Thread(ReceiveMessages);
    receiveThread.IsBackground = true;
    receiveThread.Start();
}
```

Рис. 3.7. Логіка конекту клієнту до северу (з боку клієнту)

3.4.2. Отримання даних про використання ресурсів сервісами

Основною задачею клієнта є перехоплення точних даних щодо ресурсо-споживання процесів.

Оскільки додаток працює у багато поточному режимі, а дані, отримані на клієнті використовуються одночасно декількома службами (логування, передачі та,

у випадку серверу - відображення) необхідно забезпечити уникнення ситуації DataRace.

Для цього застосовується принцип неблокуючих черг. Всі дані спочатку зарисуються в спеціальні потоко-безпечні структури даних (System.Collections.Concurrent) (рис. 3.).

```
for(; ; )
{
    if(concurrentQueue.Count > 20)
        concurrentQueue.TryDequeue(out long t);

    concurrentQueue.Enqueue(CPUCounter.PrivateMemorySize64);
    UpdChart(_cpuChart);
    Thread.Sleep(tim);
}
```

Рис. 3.8. Отримання даних про використання процесом PrivateMemory

3.4.3. Організація підключення плагінів

Для того щоб динамічна бібліотека оброблялась як модуль розширення для"NetServices Easy Monitor", вона повинна реалізовувати та експортувати декілька функцій, тобто реалізувати відкритий інтерфейс (Додаток Г).

3.4.4. Створення інтерфейсу

Інтерфейс програми створювався за допомогою Windows Presentation Foundation (Wpf) - аналогу WinForms, системи для створення додатків з графічним інтерфейсом під ОС Windows. (рис. 3.9.)

```
<Grid Grid.Column="1">
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="20px"/>
        <RowDefinition />
    </Grid.RowDefinitions>
    <Label Name="ChartLabel" Content="{Binding Path=LabelTxt}" Grid.Row="0"/>
    <Grid x:Name="ChartGrid1" Grid.Row="1"/>
</Grid>
```

Рис. 3.9. Приклад розмітки графічного інтерфейсу

Для розбиття всієї робочої області використовується елемент <Grid>

Для зручного використання діаграм використовуються розширення LiveCharts.Wpf.

Також активно застосовувались такі елементи як Button, ListBox, Label тощо. Для прив’язки елементів інтерфейсу до змінних значень відображуваних даних використовувалась технологія Binding-у.

3.5. Складання документації до програмного модулю "NetServices Easy Monitor"

3.5.1. Структура файлів конфігурації

В системі передбачена наявність двох файлів конфігурації. Один – зі сторони клієнту(рис. 3.), інший – зі сторони серверу (рис. 3.). Обидва файли конфігурації мають формат xml. Такий вибір формату зумовлений легкістю та швидкістю серіалізації/десеріалізації таких документів програмними засобами, а також високим ступенем сприйнятливості для людини (Human Readable document).(рис. 3.10)

Конфігурація Сервера

ServerConfig.xml > ...

```
<ServerConfig>
    <serverId>6F2B15E4-627E-4E61-B9FE-A9762ABF3B08</serverId>
    <serverName>NSEMserver</serverName>
    <permits>1</permits>
    <logPath>D:\NSEM\Server\Logs\</logPath>
    <logLifeTime>90</logLifeTime>
    <logErrorLvl>0</logErrorLvl>
    <dbPath>D:\NSEM\Server\DataBase\</dbPath>
    <dbLifeTime>360</dbLifeTime>
    <templatesPath>D:\NSEM\Server\Templates\</templatesPath>
    <Server>localhost</Server>
    <Port>8001</Port>
    <retryCount>15</retryCount>
</ServerConfig>
```

Рис. 3.10 Приклад файлу конфігурації сервера

<serverId> - GUID серверу. Необхідний для точної ідентифікації серверу NSEM під час мережної комунікації.

<serverName> - ім’я серверу. Необхідне для спрощеної ідентифікації серверу NSEM оператором-людиною. Задається строковим значенням.

<permits> - рівень прав доступу серверу до налаштувань клієнтів. За замовчуванням встановлено значення 1(всі права активовано). Також встановлюється значення 0 (тільки читання).

<logPath> - аболютний шлях до внутрішніх логів серверу.

<logLifeTime> - максимальний час локального зберігання одного логу серверу. Задається цілим невід’ємним числом, яке відображує кількість днів збереження.

<logErrorLvl> - рівень деталізації внутрішнього логу. Приймає такі значення: 3 – тільки критичні помилки (Error level); 2 – попередження (Warning level); 1 – інформація мережної комунікації (Info level); 0 – всі системні повідомлення (Debug level). Кожний наступний рівень включає в себе повідомлення попередніх. За замовченням має значення 3.

<dbPath> - абсолютний шлях до БД серверу.

<dbLifeTime> - максимальний час локального зберігання БД серверу. Задається цілим невід’ємним числом, яке відображує кількість днів збереження. Значення «-1» зарезервоване, та встановлюється за замовченням – не видаляти БД.

<templatesPath> - абсолютний шлях до шаблонів звітів.

<Server> - мережева адреса серверу.

<Port> - «слухаючий» порт.

<retryCount> - кількість спроб звернення до клієнту.

Конфігурація Клієнта (рис. 3.11)

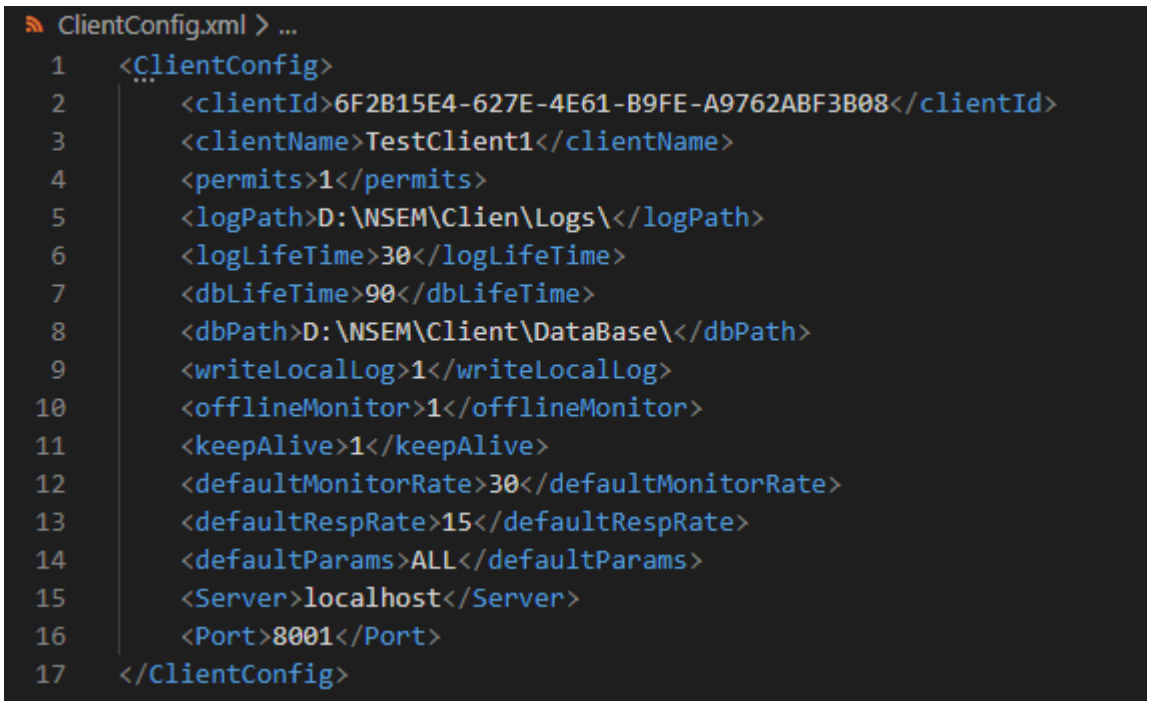

Рис. 3.11 Приклад файлу конфігурації клієнта

< clientId> - GUID клієнта. Необхідний для точної ідентифікації клієнта NSEM під час мережної комунікації.

<clientName> - ім’я клієнта. Необхідне для спрощеної ідентифікації клієнта NSEM оператором-людиною. Задається строковим значенням.

<permits> - рівень прав доступу клієнта при запитах на сервер. За замовчуванням встановлено значення 0 (тільки читання). Також встановлюється значення 1 (всі права активовано).

<logPath> - аболютний шлях до внутрішніх логів клієнта.

<logLifeTime> - максимальний час локального зберігання одного логу клієнта. Задається цілим невід’ємним числом, яке відображує кількість днів збереження.

<logErrorLvl> - рівень деталізації внутрішнього логу. Приймає такі значення: 3 – тільки критичні помилки (Error level); 2 – попередження (Warning level); 1 – інформація мережної комунікації (Info level); 0 – всі системні повідомлення (Debug level). Кожний наступний рівень включає в себе повідомлення попередніх. За замовченням має значення 2.

<dbPath> - абсолютний шлях до БД клієнта.

<dbLifeTime> - максимальний час локального зберігання БД серверу. Задається цілим невід'ємним числом, яке відображує кількість днів збереження. Значення «-1» зарезервоване - не видаляти БД.

<writeLocalLog> - визначає необхідність запису локального логу клієнта. Приймає значення: 0 –ні; 1 – так. За замовченням 1.

<offlineMonitor> - визначає необхідність моніторингу в оф-лайн режимі (при відсутності з'єднання з сервером). Приймає значення: 0 – ні; 1 – так. За замовченням 1.

<keepAlive> - визначає необхідність продовження моніторингу при обриві з'єднання з сервером. Приймає значення: 0 – ні; 1 – так. За замовченням 1.

<defaultMonitorRate> - кількість кадрів стану клієнта за хвилину, що створюються в оф-лайн режимі моніторингу. Задається цілим невід'ємним числом.

<defaultRespRate> - кількість кадрів стану клієнта, що надсилаються серверу за хвилину. Задається цілим невід'ємним числом.

<defaultParams> - шлях до шаблону кадру системи. Значення «ALL» зарезервоване та встановлюється за замовченням – моніторинг здійснюється по всім активним сервісам та всім можливим параметрам.

<Server> - мережева адреса серверу.

<Port> - «слухаючий» порт.

3.5.2. Структура Кадру стану клієнта

Шаблон Кадру стану клієнта

Для клієнта передбачена наявність шаблону Кадру стану клієнта. Шаблон має формат xml. Такий вибір формату зумовлений легкістю та швидкістю серіалізації/десеріалізації таких документів програмними засобами, а також високим ступенем сприйнятливості для людини (Human Readable document), можливістю легкого ручного конфігурування. (рис. 3.12)

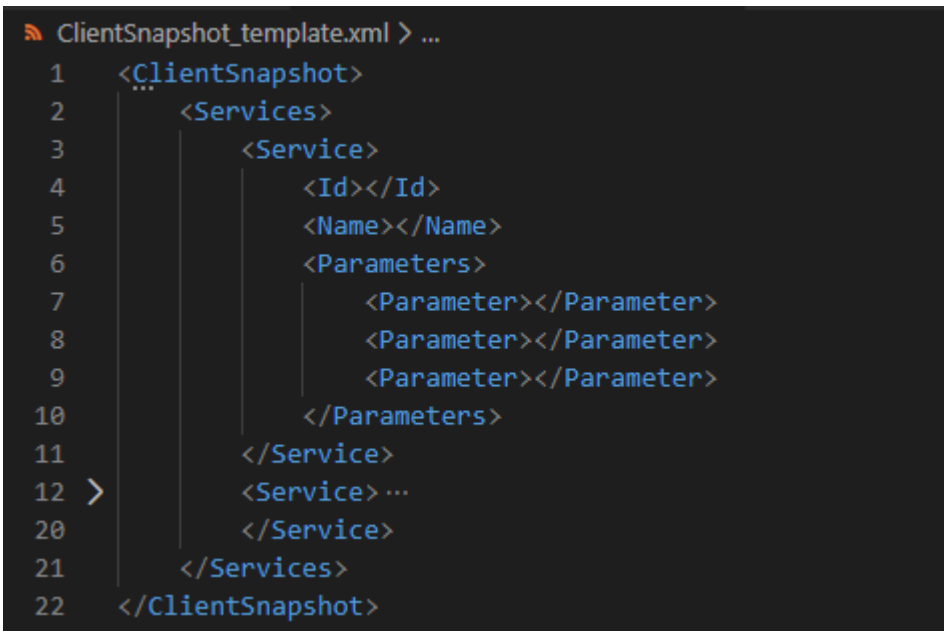

Рис. 3.12. Приклад пустого шаблону Кадру стану клієнта

Шаблон має містити перелік всіх сервісів, зазначених для моніторингу. Кожен сервіс оформлюється у форматі XML як окрема нода. Для кожного сервісу визначається:

- <Id> - ідентифікатор процесу
- <Name> - ім’я процесу
- <Parameters> - список-перелік параметрів для моніторингу

Для кожного з параметрів моніторингу <Parameter> визначається його зарезервована назва (CPU, RAM, TRAFFIC1 тощо).

Кадр стану клієнта

Кадр стану клієнта представляє собою текстовий файл, який містить повну інформацію про стан системи в певний момент часу.(рис. 3.13.)

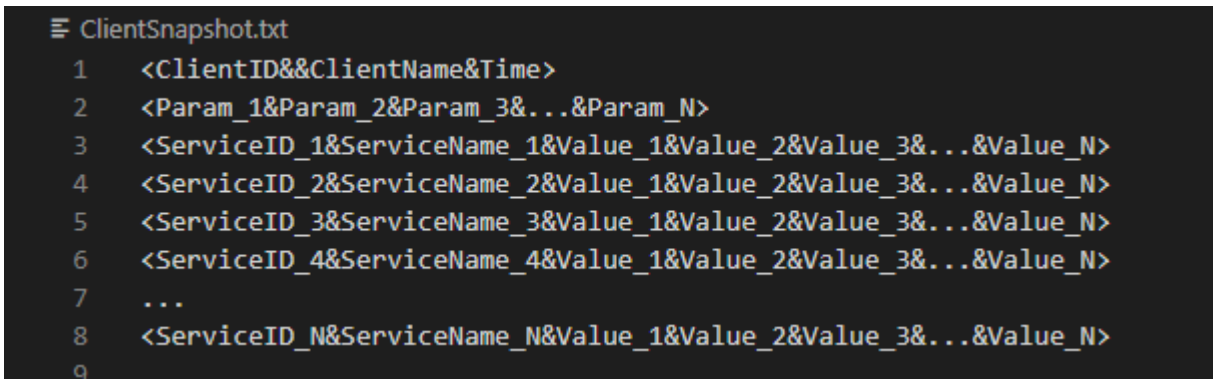

Рис. 3.13. Шаблон кадру стану клієнта в форматі TXT

Структура файлу

Файл логічно розбивається на «речення». «Речення» розміщуються між символами «<» та «>». Кожне «речення» розбивається на слова, роздільником є символ «&».

Кожне речення, окрім перших двох починається із ідентифікатору (ServiceID) та назви (ServiceName) сервісу, далі записується перелік значень параметрів (Value).

Перші два «речення» є системною інформацією:

1. <ClientID&&ClientName&Time> - послідовний запис ідентифікатору клієнта, на якому ведеться моніторинг, ім’я клієнта та точний час створення кадру.
2. <Param_1&Param_2&Param_3&...&Param_N> - маска, за якою буде розшифровуватись значення параметрів у наступних «реченнях». Якщо якийсь параметр не заданий для моніторингу для сервісу, замість нього буде записано символ «@».

Такий формат кадру дозволяє зменшити об’єм інформації при її зберіганні та передачі на сторону серверу.

3.5.3. Бази даних

БД серверу (рис. 3.14.)

База даних серверу системи NetServices Easy Monitor являє собою традиційну реляційну БД, що складається з чотирьох таблиць (рис. 3.):

1. Clients – містить перелік клієнтів, їх мережеві адреси та імена.
2. ClientSettings – містить всі налаштування, які клієнт має отримувати від серверу в процесі синхронізації та доналаштуваня.
3. Services – перелік наявних сервісів, які підлягають моніторингу.
4. Resources – таблиця у котру відбувається десеріалізація кадрів стану клієнта, тобто містить данні про ресурси, які використовуються сервісами.

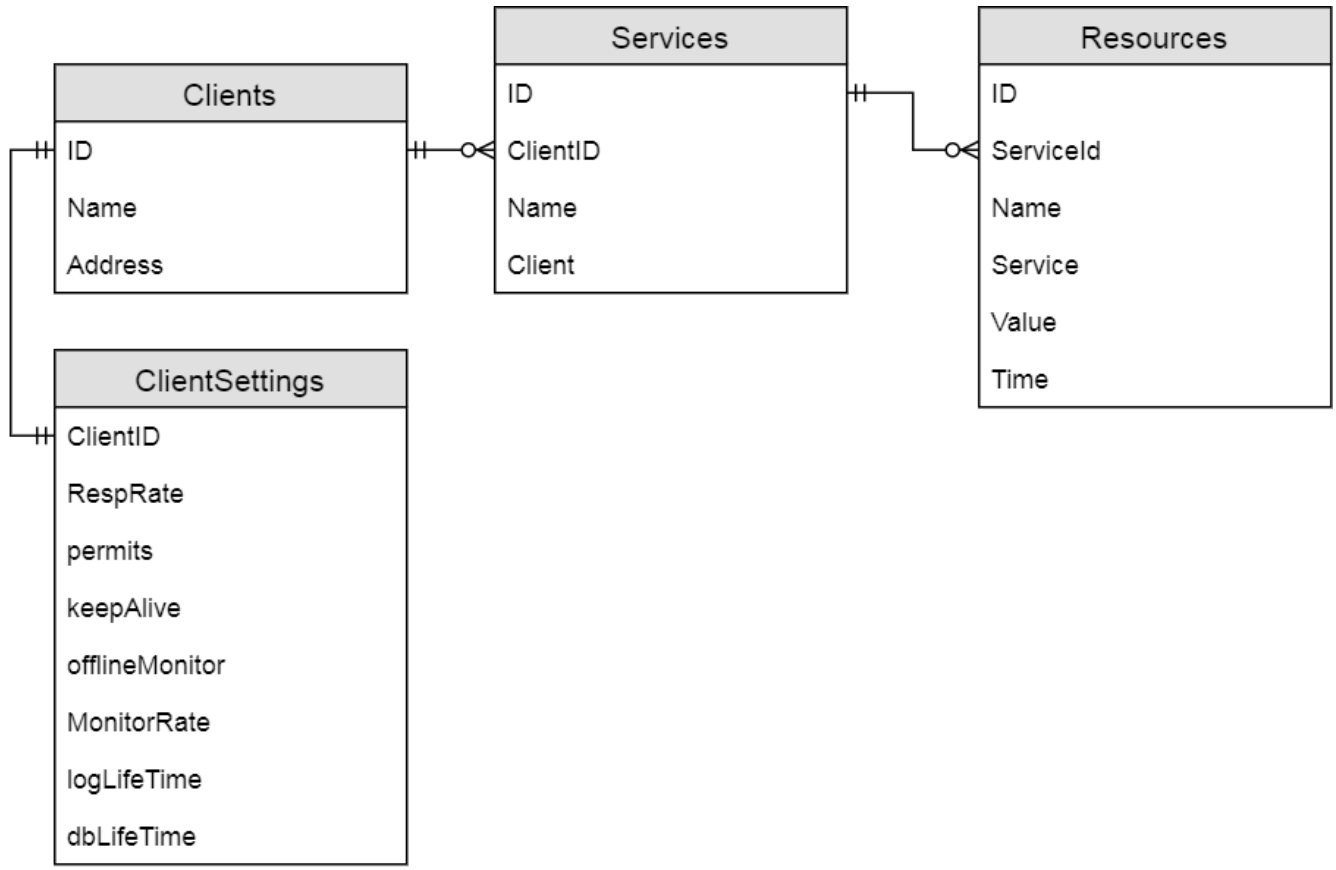

Рис. 3.14. Схема БД серверу NetServices Easy Monitor

БД клієнта (рис. 3.15**.)**

База даних клієнта є, по суті, спрощеним варіантом БД серверу – вона містить лише таблиці Services та Resources, що мають ту ж саму структуру, що й однойменні таблиці серверної БД (рис. 3).

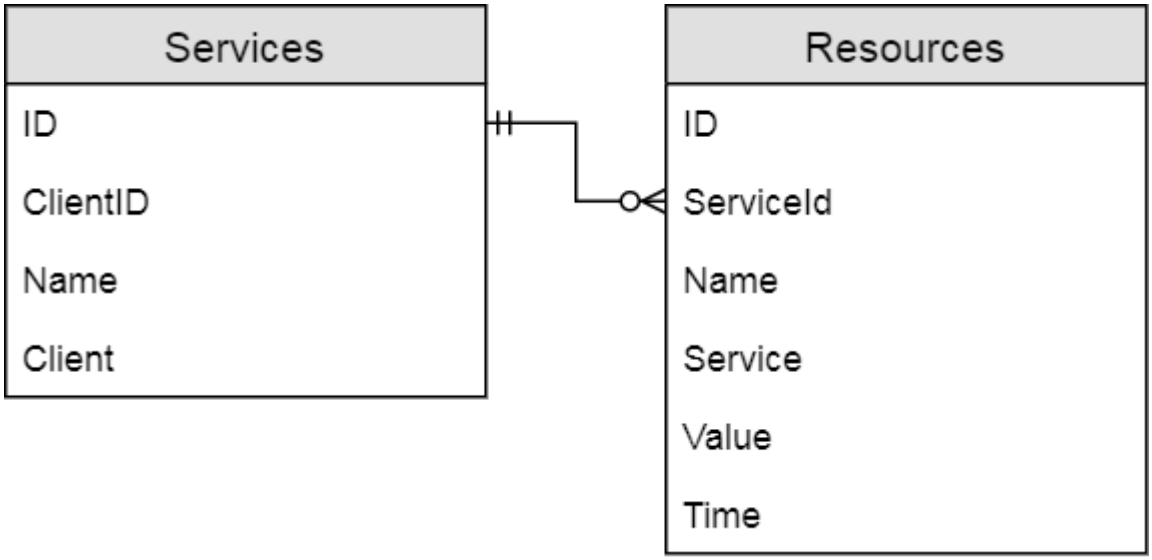

Рис. 3.15. Схема БД клієнту NetServices Easy Monitor

3.6. Огляд користувацького інтерфейсу

Графічний інтерфейс присутній лише у серверного додатку NetServices Easy Monitor Server (рис. 3.16). Це зумовлено архітектурними особливостями системи – саме сервер має у своєму розпорядженні функціонал для створення звітності про стан кліенів. Агенти моніторингу ж виконують лише функцію перехоплення необхідних даних та подальшої передачі інформації. Одною з головних вимог до агентів моніторингу є їх компактність, а складний графічний інтерфейс тягне за собою зростання вживання ресурсів. Отже, клієнтський додаток являє собою стандартну консоль.

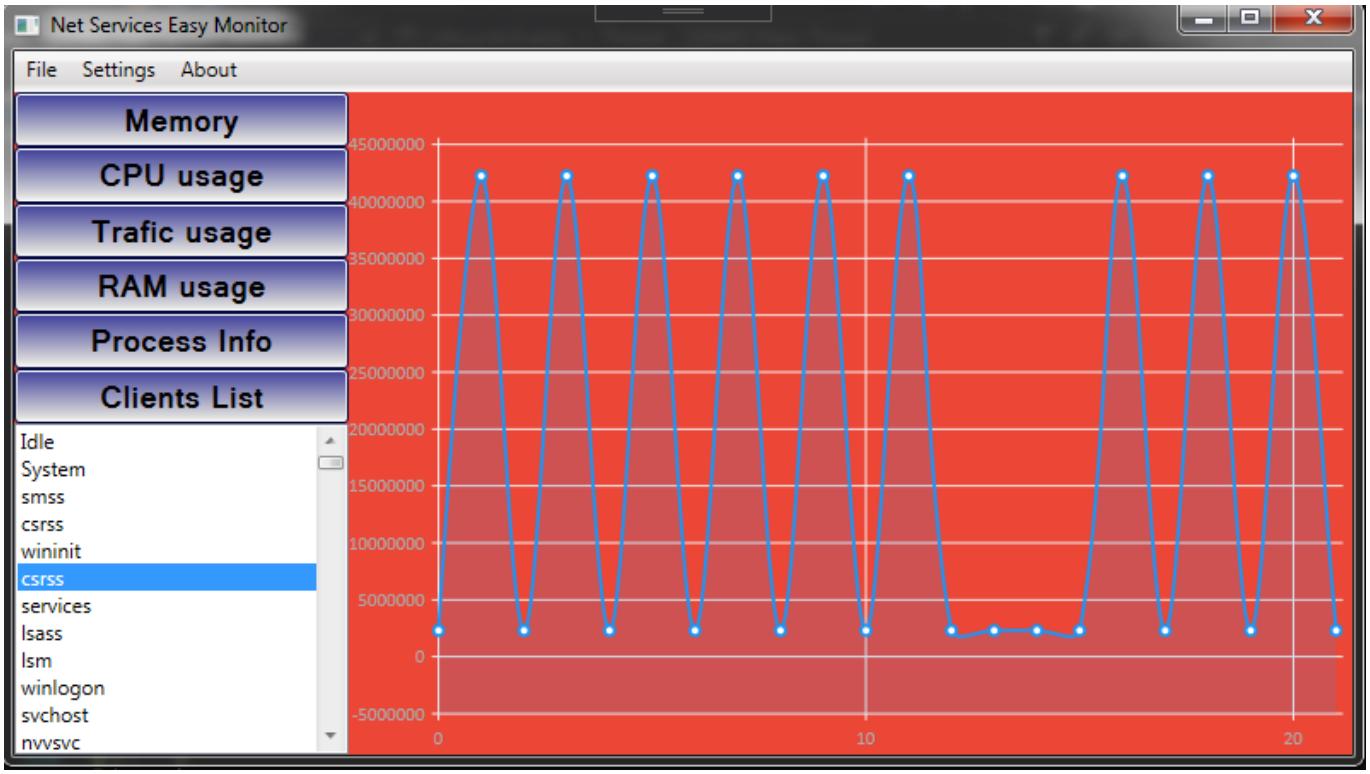

Рис. 3.16. Інтерфейс програми при запуску

Інтерфейс серверної частини системи моніторингу можна розподілити на 4 частини:

1. Головне меню (рис. 3.16.)

   Надає можливість користувачу отримати доступ до головних налаштувань програми (рис. 3.23), до діалогу відкриття та збереження звітів, інформації про версію ПЗ.

Рис. 3.17. Головне меню

2. Панель Інструментів – включає в себе перемикачі режимів відображення. (рис. 3.17.)

   Надає користувачеві перемикати «вкладинки» параметрів моніторингу, списку доступних клієнтів(рис. 3.22) та клієнтських налаштувань.

Рис. 3.18. Панель Інструментів

3. Список Активних сервісів(рис. 3.18.)

   Дозволяє обрати певний сервіс, дані з якого ми бажаємо отримати

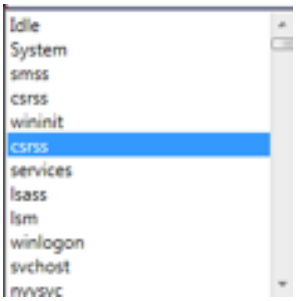

Рис. 3.19. Список Активних сервісів

4. Головне вікно відображення (рис. 3.19.)

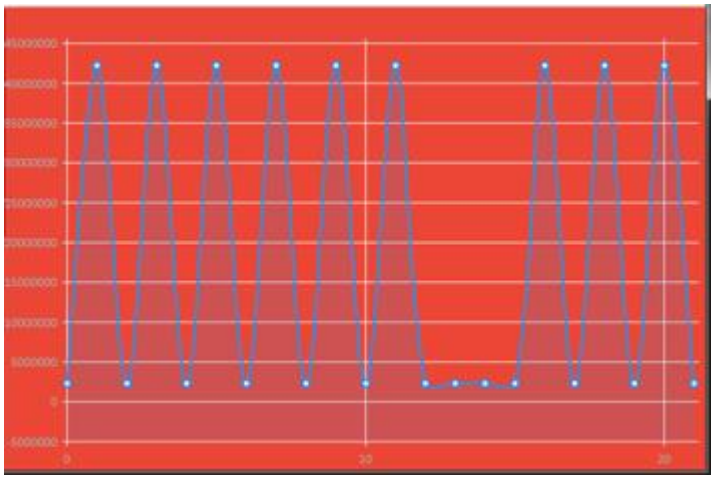

Рис. 3.20. Головне вікно відображення

Зі старту на головній сторінці відображені налаштування серверу (рис. 3.16)

В режимі Real-time моніторингу в головному вікні графічно відображаються показники моніторингу параметрів з обраної категорії (рис. 3.21).

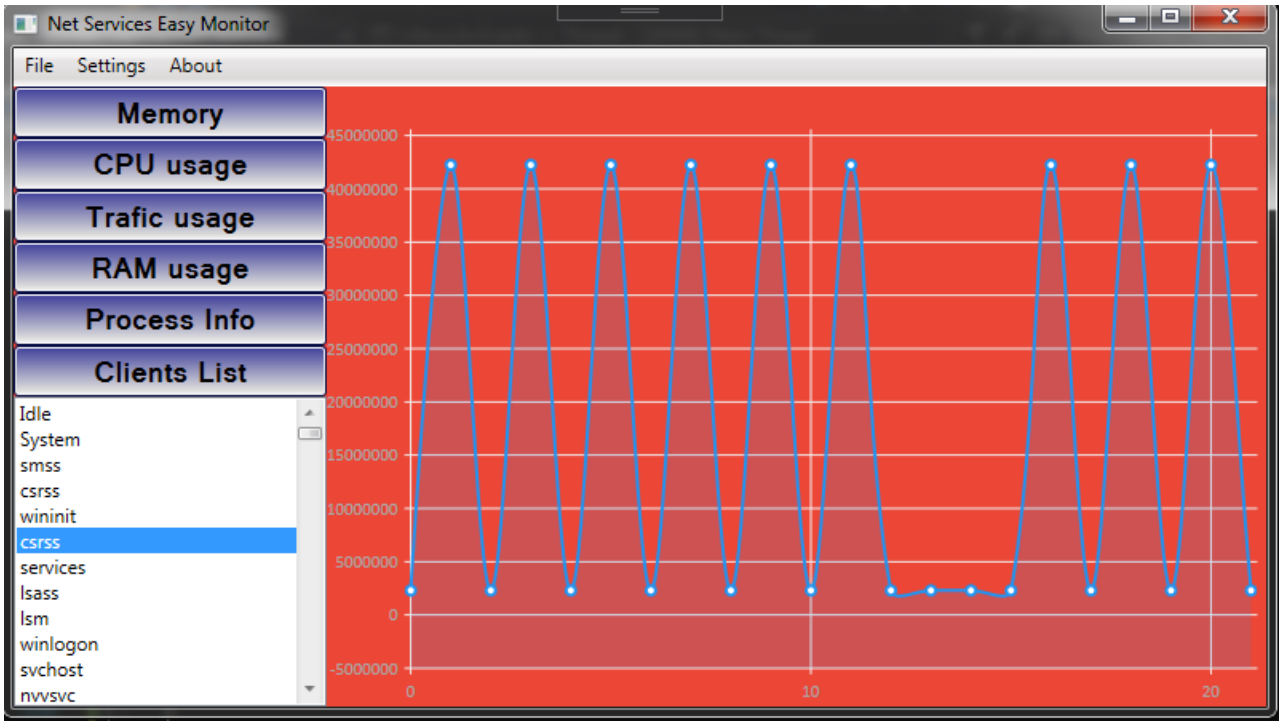

Рис. 3.21. Інтерфейс програми при запуску моніторингу параметру

Для перемикання перегляду параметрів роботи певного клієнту – треба обрати його зі списку активних агентів (Рис. 3.22).

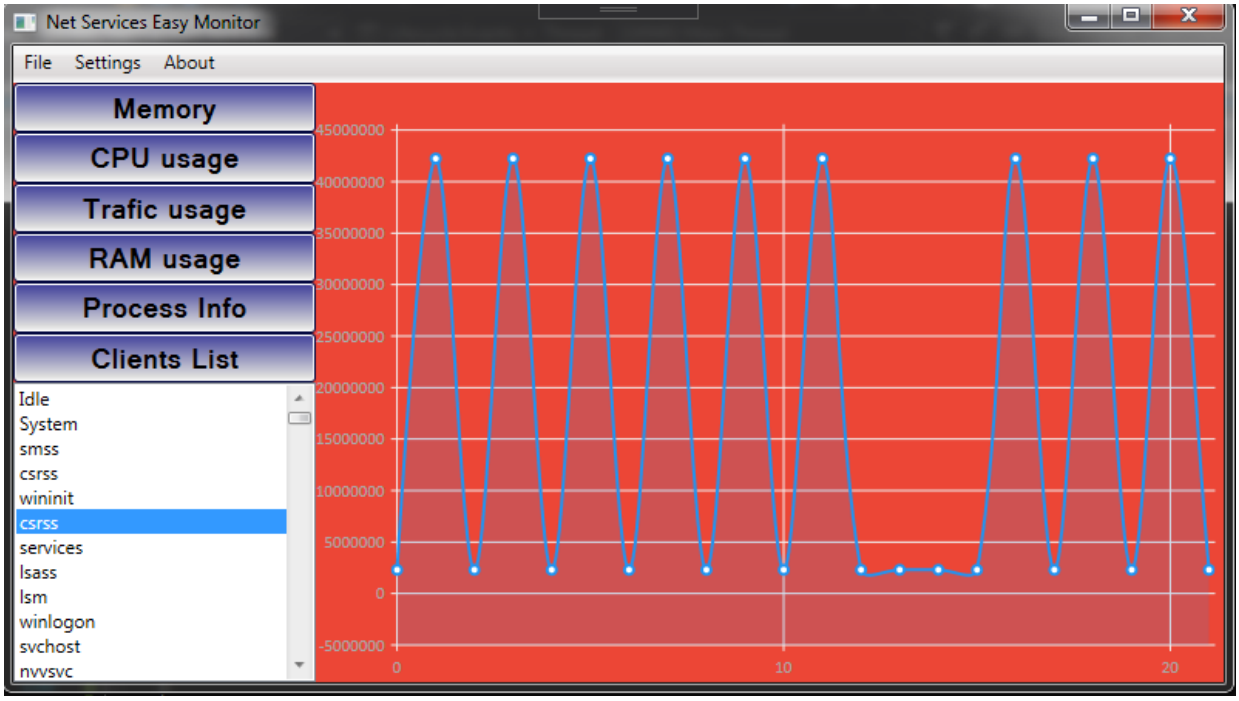

Рис. 3.22. Вибір клієнту

В пункті меню «Client Settings» можна обрати сервіси для моніторингу та параметри для кожного з сервісу (рис. 3.23).

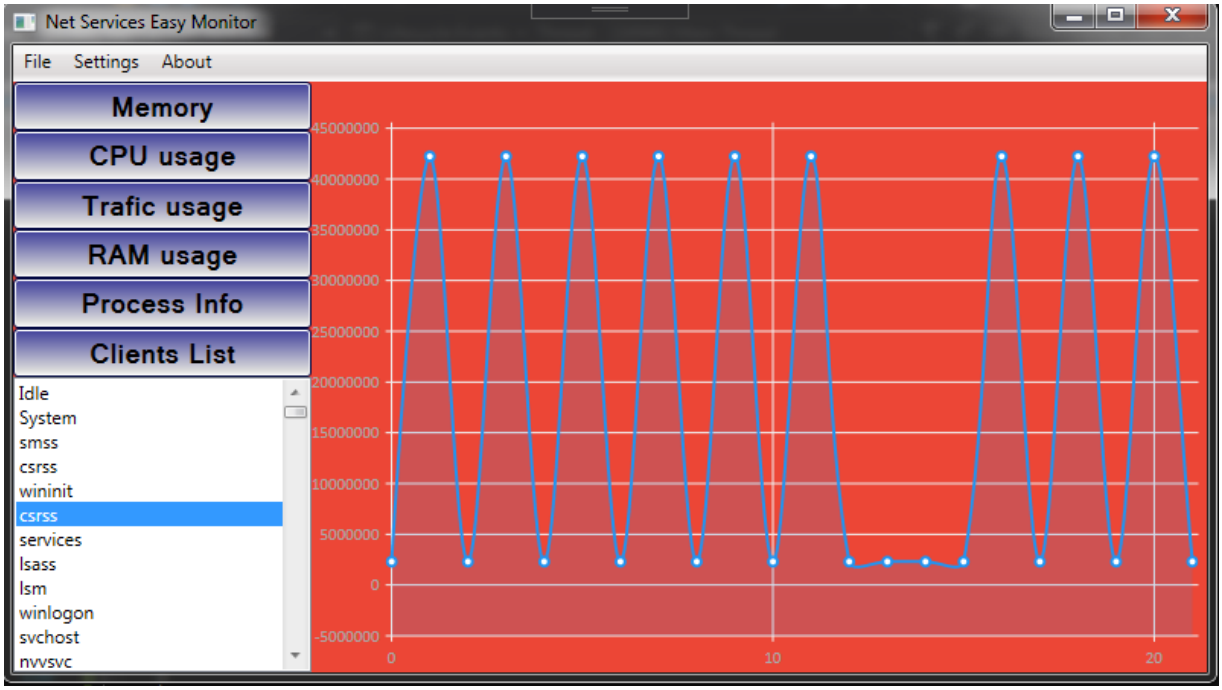

Рис. 3.23. Налаштування моніторингу клієнту

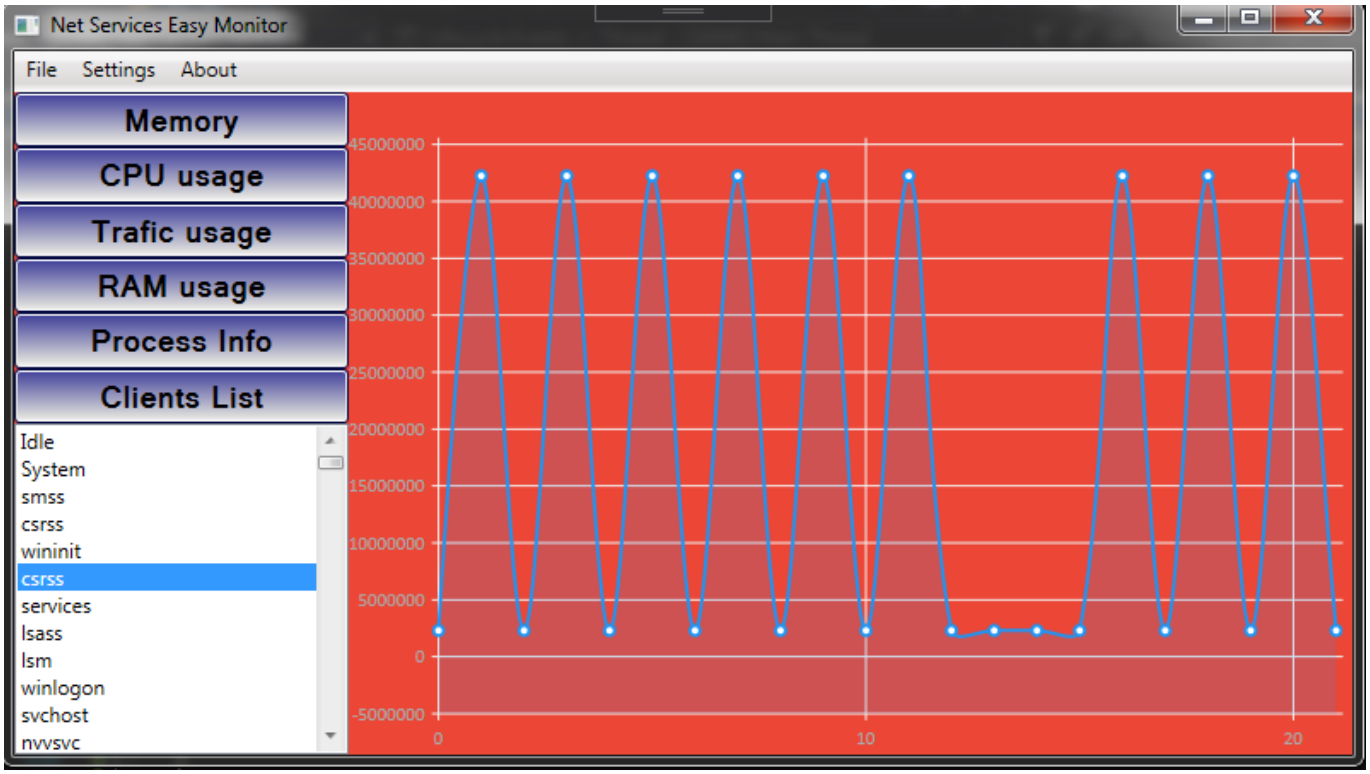

Рис. 3.24. Загальні відомості про процес

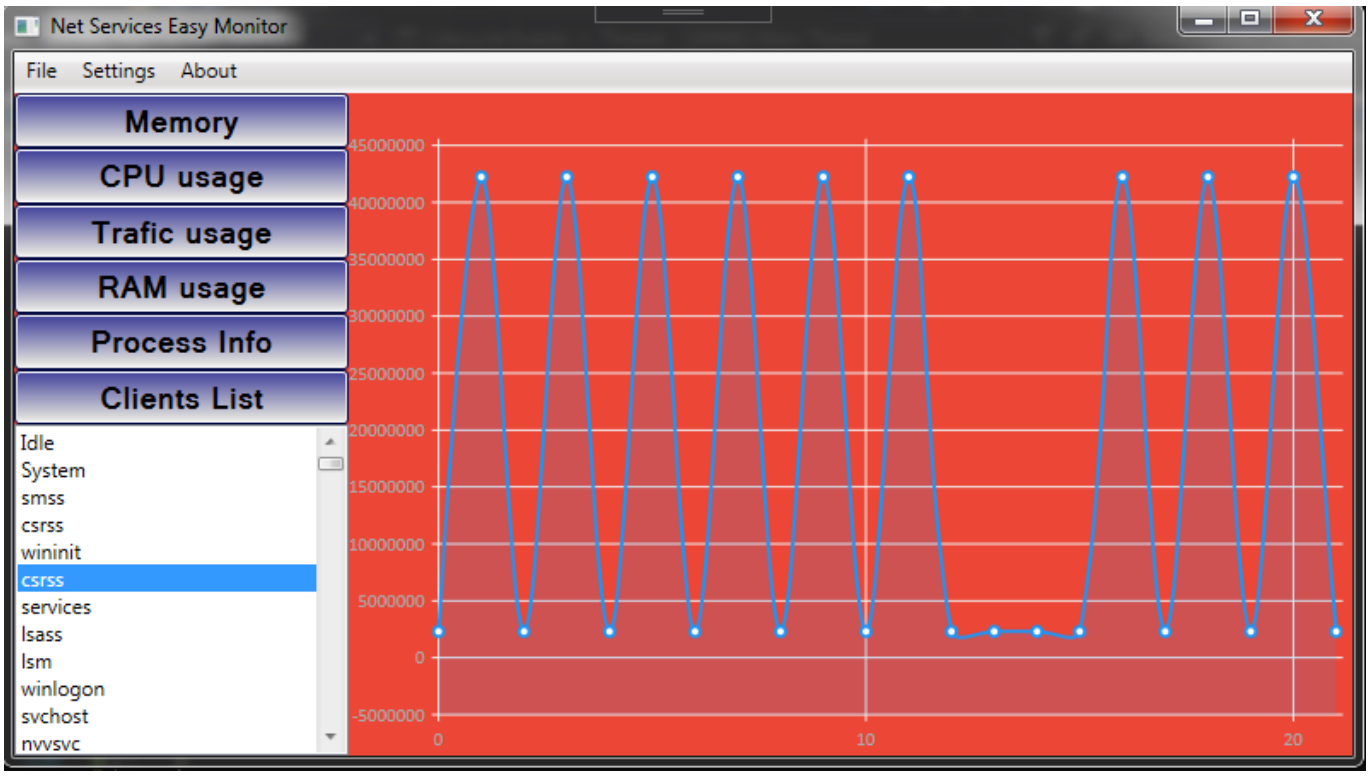

Рис. 3.25. Інтерфейс клієнту

Висновки до розділу 3

В процесі мережевого адміністрування часто з'являється необхідність в мікро-контролі ресурсів, споживаних клієнтськими сервісами. Однак, такий тип моніторингу є досить ресурсномістких, і як правило, входить до складу великих приватних корпоративних рішень недоступних для пересічних Користувачів.

В ході роботи було розроблено програму, яка здійснює детальний моніторинг як апаратних, так і мережевих ресурсів, споживаних сервісами та додатками, запущеними на стороні клієнтів.

Окрім безпосередньо програми, в ході розробки було спроектовано дві бази даних для систематизованого збереження результатів моніторингу; розроблено формат збереженя миттєвих даних стану клієнту – Кадр стану клієнта. Розроблений формат кадру дозволяє зменшити об’єм інформації при її зберіганні та передачі на сторону серверу.

Розроблене програмне рішення надає можливість:

- спостерігати стан сервісів в мережі в режимі реального часу
- Запитувати статистику роботи клієнтів / додатків / сервісів за період часу
- Створювати звітність по роботі з мережею, конфігурувати тип звітності
- Створювати на основі наданого API плагіни, для розширення базового функціоналу - створювати власні види звітів, генерувати сигнали тощо.

Програмне рішення являє собою дві компоненти:

1. сервер, який здійснює менеджмент клієнтських підключень, збір статистики і конфігурацію звітів
2. клієнт, який моніторить локальний стан додатків і сервісів, що збирає локальну статистику роботи системи і забезпечує зв'язок з сервером моніторингу.

Кожна з компонент може бути налаштована завдяки XML-файлам конфігурації.

Графічний інтерфейс присутній лише у серверного додатку NetServices Easy Monitor Server. Клієнтський додаток являє собою лише стандартну консоль.

Інтерфейс серверної частини системи моніторингу можна розподілити на 4 частини:

1. Головне меню
2. Панель Інструментів – включає в себе перемикачі режимів відображення.
3. Список активних сервісів
4. Головне вікно відображення

ВИСНОВКИ

Сьогодення диктує жорсткі вимоги щодо підтримання задовільної якості роботи розподілених мереж, а отже й до спеціалізованого програмного забезпечення.

Зростання складності мереж ускладнює процес їх адміністрування, який включає в себе такі завдання, як:

- пошук та усунення несправностей апаратної частини;
- установка нових служб і протоколів;
- налаштування програмного забезпечення;
- пошук шляхів оптимізації роботи мережі.

При адмініструванні мереж першочерговим завданням є своєчасне реагування на виникаючі проблеми і їх усунення в мінімально можливі терміни. Для оперативного і ефективного вирішення завдань адміністрування необхідна достовірна інформація про роботу компонентів і систем мережі. Ця інформація може бути зібрана і оброблена, зокрема, за допомогою спеціальних програмних систем моніторингу.

Моніторинг - це збір, обробка, агрегування і відображення кількісних даних в реальному часі про систему, таких як кількість запитів і типи запитів, кількість і типи помилок, час обробки, час життя і використання ресурсів сервера / клієнта / додатки / сервісу.

В ході дослідження і аналізу сучасних підходів до організації систем моніторингу мережі було виявлено, що такі системи мають відповідати наступним вимогам:

- прийнятна швидкість роботи;
- незначне додаткове навантаження на ресурси системи (в тому числі додатковий службовий трафік);
- інтуїтивний інтерфейс (легкість використання незалежно від кваліфікації оператора);
- портативність;

- масштабованість;
- наявність можливості конфігурації.

Наразі існує безліч різних інструментів для здійснення моніторингу, проте вони доволі громіздкі, вимагають високої кваліфікації від операторів та спеціалізуються, здебільшого на аналізі мережевого трафіку та стану комунікаційного обладнання. Отож на ринку одібного програмного забезпечення існує попит на системи з наступним функціоналом:

Виходячи з визначених вимог до систем моніторингу, було прийнято рішення реалізувати наступний функціонал:

- Моніторинг стану хостів: (завантаження процесора, використання диска, системні логи).
- Отримання деталізованої інформації по миттєвому (або за певний період) споживання таких ресурсів, як: завантаженість процесора, оперативної пам'яті, відеопам'яті, споживання зовнішньої пам'яті, трафік. Інформація про дані о параметрах може бути запрошена пакетно, або по кожному параметру окремо.
- Моніторинг мережевих служб: (SMTP, POP3, HTTP, NNTP, ICMP, SNMP)
- Логування, створення звітів
- Інструмент створення звітів за певний період часу а також отримання миттєвої статистики в режимі реального часу по кожному з клієнтів / сервісів.
- Можливості для масштабування: Проста архітектура модулів розширень (плагінів) для полегшання процесу розробки додаткових способів перевірки служб;
- Можливість визначати обробники подій, що відбулися зі службами або хостами для проактивного вирішення проблем.

Виходячи із виявлених вимог до систем моніторингу запропоновано метод моніторингу диференційованого вживання ресурсів окремими сервісами у розподілених системах.

Запропонований метод включає в себе:

- Новий формат збереження інформації, отриманої з монітору агента – Кадр стану клієнта.
- Кастомний спосіб диференційного збереження інформації моніторингу на агентах задля зменшення навантаження на ресурси клієнта.

На основі запропонованого методу була розроблена система «NetServices Easy Monitor».

Розроблена система слугує інструментом для забезпечення стабільної роботи розподіленої мережі:

- Своєчасного виявлення відмов, перебоїв та суттєвих відхилень показників продуктивності мережі.
- Прогнозування нестабільності в роботі мережі на основі даних про стан обчислювальних та мережевих ресурсів мережі.
- Створення звітів та зручної інфографіки щодо поточного стану системи
- Створення бази даних щодо показників роботи мережі, яку можна використовувати для подальшого аналізу.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ТА ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Таненбаум Э., Уэзеролл Д. - Компьютерные сети. 5-е изд. – СПб.: Питер, 2012. – 960 с.
2. Олифер В.Г., Олифер Н.А. - Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов. 5-е изд. – СПб.: Питер, 2016, – 992 с.
3. Christian Cachin, Rachid Guerraoui, Luís Rodrigues - Introduction to Reliable and Secure Distributed Programming. Springer; 2nd ed. 2011 – 367 pages
4. Mike Julian - Practical Monitoring: Effective Strategies for the Real World. O'Reilly Media; 1ie ed. 2017 - 170 pages
5. Столлингс В. Современные компьютерные сети. 2-е изд. – СПб.: Питер, 2003. – 783 с.
6. Brendan Burns - Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media; 1ie ed. 2018. - 166 pages
7. Сергеев А. Н. - Основы локальных компьютерных сетей. 1-2 изд. – Лань, 2016 – 184 с.
8. Ed Wilson - Network Monitoring and Analysis: A Protocol Approach to Troubleshooting. Prentice Hall 2000 - 350 pages
9. Dr. Moustafa Elshafei - Modern Distributed Control Systems: A comprehensive coverage of DCS technologies and standards. CreateSpace Independent Publishing Platform; 1ie ed. 2016 - 478 pages
10. David Makofske, Michael J. Donahoo, Kenneth L. Calvert - TCP/IP Sockets in C#: Practical Guide for Programmers (The Practical Guides). Morgan Kaufmann; 1st ed. 2004 - 175 pages
11. Sean Burns - Hands-On Network Programming with C# and .NET Core: Build robust network applications with C# and .NET Core. Packt Publishing; 1st ed. 2019 - 488 pages
12. Шилдт, Герберт C# 4.0. Полное руководство / Герберт Шилдт. - М.: Вильямс, 2015. - 670 с.

13. Дэвис, Алекс Асинхронное программирование в C# 5.0 / Алекс Дэвис. - М.: ДМК Пресс, 2015. - 120 с.

14. Valeriy Vyatkin - IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, Instrumentation Society of America, USA. 2nd ed. 2011 - 244 pages

15. https://docs.microsoft.com/

16. https://ru.wikipedia.org/

17. https://studopedia.com.ua/

18. https://www.zabbix.com/

19. https://www.ittsystems.com/nagios-vs-icinga-comparison/

ДОДАТОК А

Приклад файлу конфігурації сервера

```
<ServerConfig>
   <serverId>6F2B15E4-627E-4E61-B9FE-A9762ABF3B08</serverId>
   <serverName>NSEMserver</serverName>
   <permits>1</permits>
   <logPath>D:\NSEM\Server\Logs\</logPath>
   <logLifeTime>90</logLifeTime>
   <logErrorLvl>0</logErrorLvl>
   <dbPath>D:\NSEM\Server\DataBase\</dbPath>
   <dbLifeTime>360</dbLifeTime>
   <templatesPath>D:\NSEM\Server\Templates\</templatesPath>
   <Server>localhost</Server>
   <Port>8001</Port>
   <retryCount>15</retryCount>
</ServerConfig>
```

ДОДАТОК Б

Приклад файлу конфігурації клієнта

```
<ClientConfig>
   <clientId>6F2B15E4-627E-4E61-B9FE-A9762ABF3B08</clientId>
   <clientName>TestClient1</clientName>
   <permits>1</permits>
   <logPath>D:\NSEM\Clien\Logs\</logPath>
   <logLifeTime>30</logLifeTime>
   <dbLifeTime>90</dbLifeTime>
   <dbPath>D:\NSEM\Client\DataBase\</dbPath>
   <writeLocalLog>1</writeLocalLog>
   <offlineMonitor>1</offlineMonitor>
   <keepAlive>1</keepAlive>
   <defaultMonitorRate>30</defaultMonitorRate>
   <defaultRespRate>15</defaultRespRate>
   <defaultParams>ALL</defaultParams>
   <Server>localhost</Server>
   <Port>8001</Port>
</ClientConfig>
```

ДОДАТОК В

Приклад кадру стану пристрою

```
<Snapshot>
   <ClientId>6F2B15E4-627E-4E61-B9FE-A9762ABF3B08</ClientId>
   <ClientName>TestClient1</ClientName>
   <DateTime>12.07.2019 13:38:23</DateTime>
   <Services>
      <Service>
         <Id>9701</Id>
         <Name>chrome.exe</Name>
         <Parameters>
            <Parameter>
               <Name>RAM</Name>
               <Value>30806</Value>
            </Parameter>
            <Parameter>
               <Name>MEMORY</Name>
               <Value>3684</Value>
            </Parameter>
         </Parameters>
      </Service>
      <Service>
         <Id></Id>
         <Name></Name>
         <Parameters>
            <Parameter></Parameter>
            <Parameter></Parameter>
            <Parameter></Parameter>
         </Parameters>
      </Service>
   </Services>
</Snapshot>
```

ДОДАТОК Г

Інтерфейс (API) підключення плагінів

ДОДАТОК Д

Код програми