COMPUTER SYSTEMS AND NETWORKS

Vol. 5, No. 1, 2023

DOI: https://doi.org/10.23939/csn2023.01.008

УДК 004.7; 004.8

АВТОНОМНА ДЕЦЕНТРАЛІЗОВАНА СИСТЕМА МОНІТОРИНГУ КОМП'ЮТЕРНОЇ МЕРЕЖІ НА ОСНОВІ ПРОГРАМНИХ АГЕНТІВ

О. Ю. Бочкарьов

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин
E-mail: oleksii.y.bochkarov@lpnu.ua

Розглянуто проблему моніторингу комп'ютерної мережі за умов обмежень на використання системних ресурсів та високих вимог до живучості системи моніторингу. Розроблено автономну децентралізовану систему моніторингу комп'ютерної мережі, яка складається з колективу програмних агентів. Кожний агент може працювати в двох режимах: основному режимі та режимі консолі управління системою моніторингу. В основному режимі агент збирає інформацію про компэютерну мережу. В режимі консолі управління агент надає користувачу доступ до зібраної усіма агентами інформації та дозволяє виконувати команди управління системою моніторингу.

Розроблена система моніторингу дає змогу отримувати достовірнішу інформацію про роботу мережі з більшою оперативністю в умовах заданих користувачем обмежень на використання обчислювальних та мережевих ресурсів. Автономна система моніторингу побудована на основі концепції багатоагентних систем, в межах якої окремий програмний агент системи володіє певною ініціативою щодо планування та реалізації сценаріїв моніторингу. В роботі програмних агентів реалізовано методи організації адаптивних процесів збирання інформації з використанням принципів самоорганізації та концепції структурної адаптації.

Запропоновано децентралізовану програмну архітектуру автономної системи моніторингу, в якій відсутній центр управління. За рахунок цього забезпечується висока надійність та живучість системи моніторингу. В програмній архітектурі автономної системи моніторингу реалізовано прикладний програмний інтерфейс SMA та відповідну програмну бібліотеку, яка дає змогу збирати статистичні дані про роботу комп'ютерної мережі та її вузлів. Розглянуто реалізацію програмного агента та консолі управління автономної системи моніторингу комп'ютерної мережі.

Ключові слова: моніторинг комп'ютерної мережі; автономна система; децентралізоване управління; програмний агент

Вступ

Нині більшість технологій та програмних засобів моніторингу комп'ютерних мереж потребують безпосередньої участі системних адміністраторів у плануванні сценаріїв моніторингу, плануванні та проведенні натурних експериментів для визначення робочих характеристик мережі та здійснення управління мережевим обладнанням і окремими вузлами мережі [1–5]. Внаслідок цього для підтримання комп'ютерної мережі будь-якого масштабу в робочому стані потрібні високооплачувані спеціалісти, обізнані зі складним апаратним та програмним забезпеченням, що використовується для вирішення завдань моніторингу. До того ж така організація моніторингу характеризується низькою оперативністю, і, як наслідок, низькою достовірністю отримуваної інформації про роботу мережі, оскільки рішення із управління системою моніторингу приймає системний адміністратор у відповідному масштабі часу, тоді як швидкість змін у сучасних комп'ютерних мережах набагато перевищує природні можливості людини.

Іншою проблемою є наявність обмежень на кількість обчислювальних та мережевих ресурсів, що використовуються програмною системою моніторингу. Якщо система моніторингу не буде дотримуватись цих обмежень, вона може задіяти значні системні ресурси і вимірюватиме, здебільшого, результати власної роботи, тобто буде порушено принцип невтручання в об'єкт дослідження. В сучасних програмних системах моніторингу комп'ютерних мереж ця проблема далека від остаточного вирішення. Ще однією проблемою сучасних програмних систем моніторингу комп'ютерних мереж є їхні низькі надійність та живучість внаслідок централізованої програмної архітектури. В разі виходу з ладу центральної компоненти вся система повністю перестає працювати. Відтак актуальним є завдання створення технології, яка б вирішувала зазначені вище проблеми, передусім за рахунок передавання частини повноважень системного адміністратора автономній програмній системі моніторингу та використанню децентралізованої програмної архітектури.

1. Постановка задачі

Один із найперспективніших підходів до побудови автономної децентралізованої системи моніторингу комп'ютерної мережі – використання концепції багатоагентних систем [6–10], в межах якої окремий програмний агент системи володіє певною ініціативою щодо планування та реалізації сценаріїв моніторингу, автоматизованих натурних експериментів та операцій управління комп'ютерною мережею. У межах цього підходу розглядається завдання створення нової технології та програмної системи моніторингу комп'ютерних мереж з використанням мобільних програмних агентів, яка дасть змогу оперативніше отримувати достовірнішу інформацію про роботу мережі в умовах обмежень на використання обчислювальних та мережевих ресурсів, які задав користувач. Для вирішення цього завдання доцільно використати результати досліджень в області автономних розподілених систем збирання інформації, зокрема новий клас алгоритмів самоорганізації узгоджених спільних дій колективу інтелектуальних мобільних агентів, які переводять процес збирання інформації в об'єкті дослідження на новий якісний рівень [11].

Для розв'язання поставленої задачі потрібно забезпечити:

1) реалізацію децентралізованої програмної архітектури системи моніторингу комп'ютерної мережі на основі технологій багатоагентних систем, принципів самоорганізації та концепції структурної адаптації [12];

2) мінімізацію обчислювальних та мережевих ресурсів, які використовує автономна система моніторингу, за рахунок одночасної роботи обмеженої кількості мобільних програмних агентів (як правило, меншої, ніж кількість об'єктів моніторингу), здатних за власною ініціативою переміщуватися з одного вузла мережі на інший;

3) отримання найдостовірнішої інформації про роботу мережі (збільшення кількості зібраної інформації згідно із заданим критерієм) за рахунок самоорганізації програмних агентів та забезпечення узгоджених дослідницьких дій агентів на мережевих вузлах з урахуванням поточного стану комп'ютерної мережі;

4) реалізувати можливість для системного адміністратора вибирати найпріоритетніший тип інформації про комп'ютерну мережу, що дозволить спрямовувати зусилля автономної системи моніторингу на збирання саме тієї інформації, яка цікавить користувача.

2. Програмна архітектура та функції автономної децентралізованої системи моніторингу комп'ютерної мережі

Автономна система моніторингу комп'ютерної мережі має децентралізовану програмну архітектуру та складається із колективу програмних агентів, кожний з яких здатний працювати в двох режимах (рис. 1): 1) збирання інформації про комп'ютерну мережу (standard mode); 2) надання користувачу доступу до зібраної усіма агентами інформації та здійснення команд управління системою моніторингу (manager mode). Тобто в програмній архітектурі системи моніторингу відсутній

центр управління, внаслідок чого система продовжує працювати доти, доки в робочому стані залишається хоча б один агент системи. За рахунок цього забезпечуються висока надійність та живучість автономної системи моніторингу. Реалізовані алгоритми функціонування програмних агентів дають змогу отримувати заданий об'єм інформації про роботу мережі із мінімальними витратами мережевих та обчислювальних ресурсів. Забезпечено залежність роботи програмних агентів від різних класів інформації, що цікавить користувача (наприклад, інформації з різних рівнів OSI: Ethernet, IP, Application level тощо). Програмний агент системи моніторингу (рис. 1) – це уніфікована програмна компонента, яка встановлюється локально на окремому вузлі комп'ютерної мережі. Після встановлення і запуску на виконання програмного агента мережевий вузол охоплюється автономною системою моніторингу. Основне завдання агента – збирання локальної інформації про вузол та мережу в межах специфікації RMON MIB. На вимогу іншого агента цей агент може передати зібрану їм інформацію відповідно до отриманого запиту. Кожен із агентів на вимогу користувача може тимчасово виконувати функції консолі управління системи моніторингу (manager mode), тобто надавати користувачу доступ до всієї зібраної агентами інформації та виконання команд управління системою моніторингу та комп'ютерною мережею.

У роботі програмних агентів системи моніторингу комп'ютерної мережі реалізовано методи організації адаптивних процесів збирання інформації з використанням принципів самоорганізації та концепції структурної адаптації. Це дозволяє автономній системі моніторингу отримувати достовірнішу та оперативнішу інформацію про роботу комп'ютерної мережі, завдяки чому якісно покращується процес управління мережею. Використання автономної системи моніторингу дає змогу забезпечити істотне зменшення кількості людино-годин, що витрачаються на підтримання мережі в робочому стані, за рахунок передавання частини повноважень із управління мережею програмним агентам системи моніторингу.

Автономна система моніторингу комп'ютерної мережі архітектурно ґрунтується на програмній реалізації прикладного програмного інтерфейсу Simple Monitor Agent (SMA), який розроблений та реалізований у вигляді динамічної програмної бібліотеки smapi. Ця бібліотека дає змогу збирати статистичні дані про роботу стека протоколів TCP/IP. Дані зберігаються в локальній базі даних керуючої інформації (management information base, MIB). Для заповнення і керування MIB в програмній архітектурі системи моніторингу використано програмні бібліотеки snmpapi та iphlpapi. Зокрема бібліотека iphlpapi (Internet Protocol Helper) використовується для реалізації функцій мережевого адміністрування локальної комп'ютерної мережі та комп'ютерів, що до неї входять. Ця бібліотека використовується для збирання інформації про конфігурацію мережевих налаштувань та їх зміни. Для забезпечення пошуку даних у MIB використовують функції бібліотеки snmpapi.

На основі прикладного програмного інтерфейсу SMA розроблено SNMP-агента, який є головною програмною компонентою агента системи моніторингу комп'ютерної мережі. Для цього також використано функції програмної бібліотеки snmpapi, зокрема для того щоб розроблений SNMP-агент міг надсилати та отримувати SNMP-повідомлення. Взаємодія окремого SNMP-агента з іншими агентами та агентом, який працює у режимі консолі управління (менеджера) системи моніторингу, відбувається за допомогою системного сервісу SNMP (рис. 2). Відповідно розроблений SNMP-агент можна використовувати на будь-якій системній платформі, в якій реалізований сервіс SNMP.

До функцій розробленої автономної системи моніторингу належать збирання та попередній аналіз інформації про конфігурацію та роботу комп'ютерної мережі. Інформація про конфігурацію та роботу мережі визначається у межах специфікації RMON MIB (RFC 1271, RFC 1513), зокрема збирають інформацію щодо таких груп об'єктів RMON:

1) статистика (Statistics): поточні статистичні дані про роботу мережі (дані про параметри пакетів, кількість колізій тощо);

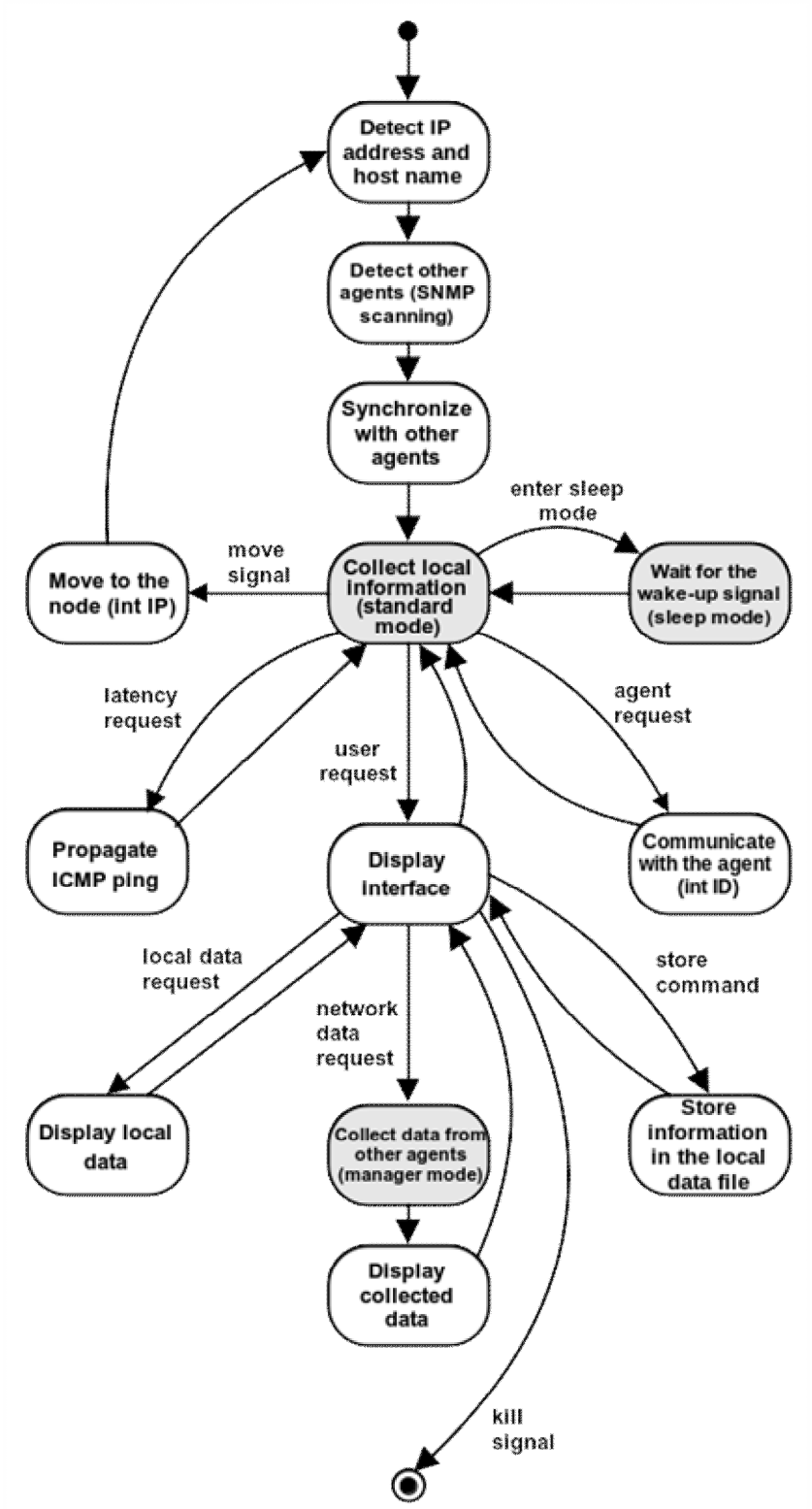

Рис. 1. Схема роботи програмного агента автономної системи моніторингу

2) історія роботи мережі (History): статистичні дані, що зберігаються через певні проміжки часу з метою аналізу тенденцій в роботі мережі;

3) мережеві вузли (Hosts): дані про мережеві вузли (MAC-адреси, IP-адреси, робочі параметри тощо);

4) найзавантаженіші мережеві вузли (HostTopN): таблиця найзавантаженіших мережевих вузлів;

5) матриця трафіку (TrafficMatrix): статистичні дані про інтенсивність трафіку між кожною парою мережевих вузлів, впорядкована у вигляді матриці.

Базовий функціонал розробленої автономної системи моніторингу відповідає стандарту ISO 7498-4, зокрема реалізовано такий базовий функціонал:

1) управління конфігурацією мережі (Configuration Management): первинне налагодження компонентів мережі, зокрема встановлення їх мережевих адрес та ідентифікаторів; управління параметрами мережевих ОС і формування топології мережі;

2) опрацювання помилок (Fault Management): виявлення проблем у мережі, повідомлення системному адміністратору про їх виникнення (на консоль адміністратора, його електронну адресу або мобільний телефон), здійснення доступних заходів для ліквідації мережневих збоїв;

3) визначення та аналіз продуктивності (Performance Management): накопичення статистичної інформації про роботу мережі, самостійна оптимізація продуктивності мережі на основі аналізу цієї інформації; оцінка пропускної здатності мережі (середньої, миттєвої, максимальної) та інтенсивності потоків даних; визначення найзавантаженіших ділянок мережі, часу відгуку та затримок передавання пакетів.

Розроблена автономна децентралізована система моніторингу комп’ютерної мережі є одним з наступних кроків у реалізації концепції автономних обчислень (Autonomic Computing) [13, 14], яка покликана перекласти основну частину повноважень із управління комп’ютерними системами з системних адміністраторів та іншого обслуговуючого персоналу на самі автономні комп’ютерні системи. Завдяки високому рівню складності сучасних комп’ютерних систем і його подальшому зростанню підтримка таких систем в робочому стані потребує значно більше витрат, ніж на їх придбання та впровадження. Відтак розроблена автономна система моніторингу дасть змогу істотно зменшити ці витрати та має великі перспективи подальшого вдосконалення і використання під час побудови інфраструктури корпоративних інформаційних систем.

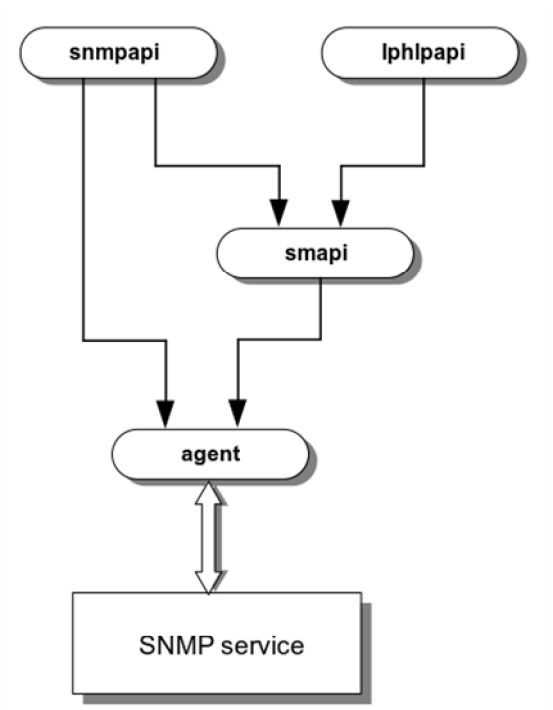

Рис. 2. Взаємодія основних програмних компонент, програмних інтерфейсів та бібліотек автономної системи моніторингу

3. Реалізація програмного інтерфейсу SMA

Прикладний програмний інтерфейс SMA реалізовано засобами мов програмування ANSCI C та C++ у вигляді динамічної програмної бібліотеки smapi, яка забезпечує збирання статистичних даних стека протоколів TCP/IP і зберігає їх в своїй локальній базі керуючих даних (MIB). Доступ до цих даних здійснюється за допомогою двох функцій, які експортує ця бібліотека (рис. 3): Set_OidPrefix (встановлює префікс ідентифікатора об'єкта) і ResolveVarBind (здійснює доступ до конкретного об'єкта бази даних).

База керуючих даних (MIB) являє собою структуру MIB_ENTRY:

```
typedef struct mib_entry {
    AsnObjectIdentifier Oid;
    void *  Storage;
    BYTE   Type;
    UINT    Access;
    UINT    (*MibFunc)( UINT, struct mib_entry *, RFC1157VarBind * );
    struct mib_entry *  MibNext;
} MIB_ENTRY;
```

де Oid – ідентифікатор об'єкта; Storage – покажчик на змінну, в якій зберігаються дані; Type – тип змінної; Access – тип доступу до змінної (читання або читання – запис); MibFunc – покажчик на функцію, що модифікує або читає дані змінної, на яку вказує покажчик цієї структури; MibNext – покажчик на наступну структуру.

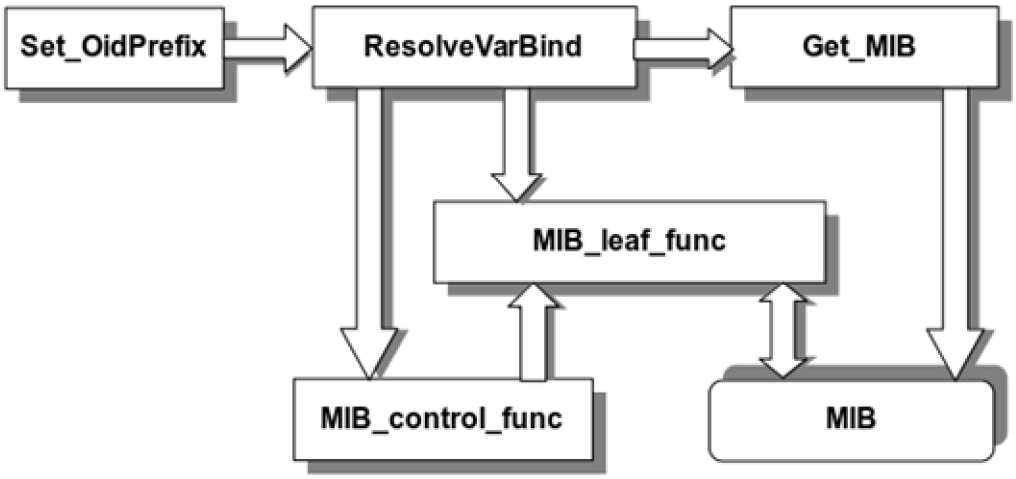

Рис. 3. Структура програмної реалізації інтерфейсу SMA (smapi)

Функція ResolveVarBind здійснює доступ до конкретного об'єкта бази даних:

```
__declspec(dllexport)  UINT ResolveVarBind(
    IN OUT RFC1157VarBind *VarBind,
    IN UINT PduAction)
```

де *VarBind – покажчик на структуру RFC1157VarBind, PduAction – тип команди. Ця функція на першому етапі виконання SNMP-агента поновлює базу даних за допомогою функції Get_MIB, а потім у циклі шукає в базі даних ідентифікатор об'єкта, переданий як параметр функції. Пошук здійснюється за допомогою функції SnmpUtilOidCmp:

```
CompResult = SnmpUtilOidCmp( &VarBind->name, &TempOid );
```

Ця функція порівнює ідентифікатор, переданий функції ResolveVarBind, з ідентифікаторами, які містяться в базі даних. Якщо результат порівняння менший від нуля (CompResult<0) і тип команди не дорівнює MIB_ACTION_GETNEXT, то функція повертає результат SNMP_ERRORSTATUS_

NOSUCHNAME, який вказує на те, що цієї змінної в MIB немає. Якщо тип команди MIB_ACTION_GETNEXT, покажчику MibPtr присвоюється адреса першого елемента структури MIB_ENTRY і встановлюється тип команди MIB_ACTION_GET, після чого викликається функція, на яку вказує покажчик *MibFunc структури MIB_ENTRY:

```
nResult = (*MibPtr->MibFunc)( PduAction, MibPtr, VarBind );
```

Якщо результат порівняння ідентифікаторів дорівнює нулю (CompResult=0), це означає, що ця змінна наявна в MIB. Тоді покажчику MibPtr присвоюється адреса поточного елемента структури MIB_ENTRY, після чого викликається функція, на яку вказує покажчик *MibFunc структури MIB_ENTRY. В ситуації, коли результат порівняння більший від нуля (CompResult>0), змінна лічильнику циклу збільшується на одиницю. Вихід із циклу відбувається за умови:

```
while( MibPtr == NULL && i < MIB_num_variables )
```

де MIB_num_variables – кількість елементів структури MIB_ENTRY.

Дані зі змінних, на які вказують покажчики Storage структури MIB_ENTRY, безпосередньо модифікуються або зчитуються за допомогою функції MIB_leaf_func:

```
UINT MIB_leaf_func(
        IN UINT Action,
        IN MIB_ENTRY *MibPtr,
        IN RFC1157VarBind *VarBind)
```

де Action – тип команди, MibPtr – покажчик на структуру MIB_ENTRY, VarBind – покажчик на структуру RFC1157VarBind. Ця функція працює так. За допомогою оператора switch case визначають, який тип команди містить параметр функції Action. Якщо Action дорівнює MIB_ACTION_GETNEXT, то перевіряють, чи існує такий елемент структури:

```
if ( MibPtr->MibNext == NULL ) {
      ErrStat = SNMP_ERRORSTATUS_NOSUCHNAME;
      goto Exit;
}
```

і викликають функцію, на яку вказує покажчик MibFunc наступного елемента структури MIB_ENTRY з такими параметрами:

```
ErrStat = (*MibPtr->MibNext->MibFunc)( MIB_ACTION_GET, MibPtr->MibNext, VarBind );
```

Якщо Action дорівнює MIB_ACTION_GET, то перевіряється тип доступу:

```
if ( MibPtr->Access != MIB_ACCESS_READ &&
    MibPtr->Access != MIB_ACCESS_READWRITE ) {
      ErrStat = SNMP_ERRORSTATUS_NOSUCHNAME;
      goto Exit;
}
```

У структуру VarBind записується значення змінної, на яку вказує покажчик Storage структури MIB_ENTRY:

```
VarBind->value.asnType = MibPtr->Type;
VarBind->value.asnValue.number = *(AsnInteger *)(MibPtr->Storage);
```

Якщо Action дорівнює MIB_ACTION_SET, то перевіряється тип доступу:

```
if ( MibPtr->Access != MIB_ACCESS_READWRITE &&
     MibPtr->Access != MIB_ACCESS_WRITE ) {
      ErrStat = SNMP_ERRORSTATUS_NOSUCHNAME;
      goto Exit;
}
```

після чого відбувається перевірка типу змінної:

```
if ( MibPtr->Type != VarBind->value.asnType ) {
      ErrStat = SNMP_ERRORSTATUS_BADVALUE;
      goto Exit;
}
```

та запис даних у MIB:

```
*(AsnInteger *)(MibPtr->Storage) = VarBind->value.asnValue.number;
```

Функція MIB_control_func виконує додатковий контроль для типу команди MIB_ACTION_SET, перевіряючи значення змінної, яка має записуватися в MIB:

```
if ( MIB_TOASTER_UP > VarBind->value.asnValue.number ||
     MIB_TOASTER_DOWN < VarBind->value.asnValue.number ) {
      ErrStat = SNMP_ERRORSTATUS_BADVALUE;
      goto Exit;
}
```

При цьому також виконується перевірка: чи відповідає значення змінної value.asnValue.number заданому діапазону значень.

Функція Get_MIB зчитує статистичні дані стека протоколів TCP/IP за допомогою таких функцій:

1) GetIpStatistics – отримання статистичних даних протоколу IP;
2) GetTcpStatistics – одержання статистичних даних протоколу TCP;
3) GetUdpStatistics – отримання статистичних даних протоколу UDP;
4) GetIfEntry – отримання статистичних даних інтерфейсного рівня.

Для того, щоб ці функції були доступні SNMP-агенту, спочатку завантажується програмна бібліотека iphlpapi і визначаються адреси пам'яті, за якими розміщенні ці функції. Завантаження бібліотеки виконується за допомогою функції LoadLibrary, а адреси пам'яті визначаються за допомогою функції GetProcAddress:

```
hDLL=LoadLibrary("iphlpapi.dll");
lpGetIfEntry = (LPFNDLLFUNC1)GetProcAddress(hDLL,"GetIfEntry");
lpGetIpStatistics = (LPFNDLLFUNC2)GetProcAddress(hDLL,"GetIpStatistics");
lpGetTcpStatistics = (LPFNDLLFUNC3)GetProcAddress(hDLL,"GetTcpStatistics");
lpGetUdpStatistics =(LPFNDLLFUNC4)GetProcAddress(hDLL,"GetUdpStatistics");
```

4. Реалізація програмного агента системи моніторингу комп'ютерної мережі

Агент системи моніторингу, зокрема його основна програмна компонента – SNMP-агент, реалізовано на мові програмування ANSCI C, оскільки він виконується на рівні системного програмного забезпечення, внаслідок чого необхідно, щоб він працював дуже швидко. Агент надсилає та отримує SNMP-повідомлення за допомогою системного сервісу SNMP (рис. 4). Для цього в програмній реалізації агента (smagent) використовуються такі функції:

```
BOOL WINAPI DllMain;
BOOL WINAPI SnmpExtensionInit;
BOOL WINAPI SnmpExtensionTrap;
BOOL WINAPI SnmpExtensionQuery.
```

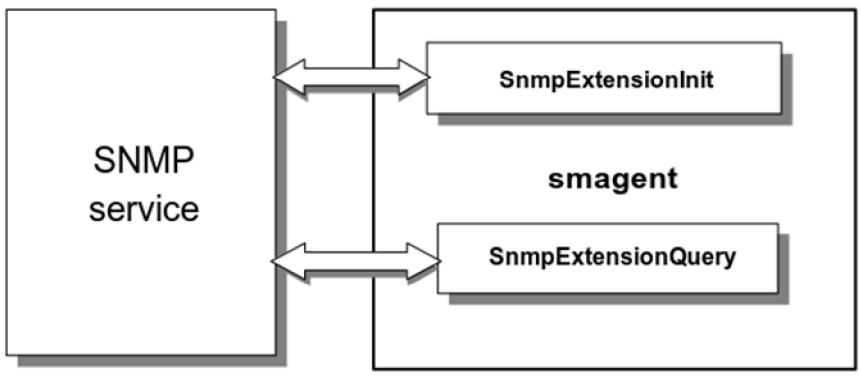

Рис. 4. Схема взаємодії агента (smagent) із системним сервісом SNMP

Функція SnmpExtensionInit реєструє у сервісі SNMP префікс ідентифікаторів об'єктів MIB. Функція SnmpExtensionQuery надсилає та отримує SNMP-повідомлення. Коли з мережі надходить SNMP-повідомлення, сервіс SNMP перевіряє у своїй базі даних префіксів-ідентифікаторів об'єктів MIB, чи за префіксом, що містить SNMP-повідомлення, зареєстрований агент. Якщо префікс SNMP повідомлення відповідає префіксу, за яким зареєстрований агент, викликається функція SnmpExtensionQuery цього агента. В функції SnmpExtensionQuery викликається функція Resolve VarBind, яка виконує обробку надісланого SNMP-повідомлення.

5. Реалізація консолі управління системи моніторингу комп'ютерної мережі

Основним завданням під час розроблення консолі управління (менеджера) системи моніторингу було створення графічного інтерфейсу користувача для відображення інформації, отриманої від агентів, та виконання команд із управління системою моніторингу. Консоль управління містить п'ять форм користувацького інтерфейсу (рис. 5) й один програмний модуль. Взаємодію консолі управління з іншими програмними компонентами системи за протоколом SNMP реалізовано за допомогою функцій стандартної бібліотеки wsnmp32.

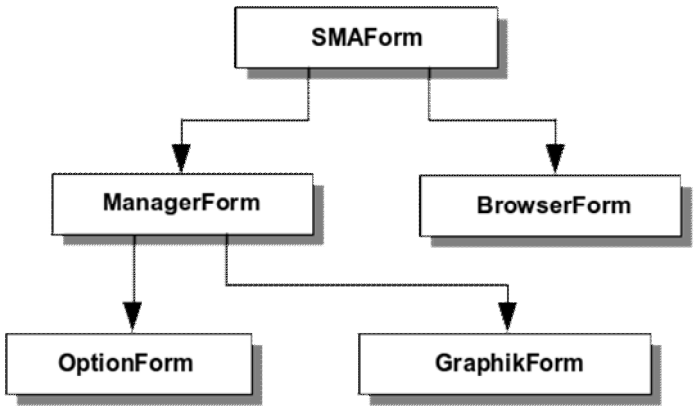

Рис. 5. Схема взаємодії форм користувацького інтерфейсу консолі управління системи моніторингу

Форма користувацького інтерфейсу SMAForm реалізує головне вікно консолі управління, з якого відбувається виклик усіх інших вікон. Форма SMAForm – це форма типу MDI, яка дає змогу створити контейнер для певної кількості інших форм типу MDI Child. На формі ManagerForm відображаються головні події моніторингу комп'ютерної мережі. Під час завантаження цієї форми, тобто в процедурі Loadform, ініціалізується сесія взаємодії за протоколом SNMP за допомогою функції SnmpCreateSession бібліотеки wsnmp32:

```
hSession = SnmpCreateSession(txSnmpListen.hWnd, WM_CHAR, 0, 0);
```

У цій функції як аргумент прив'язується процедура txSnmpListen_KeyPress, що викликається в момент надходження SNMP-повідомлень. Формування і надсилання SNMP-повідомлень виконує процедура SendRequest. SNMP-повідомлення містить такі складові: версія повідомлення, контекст (пароль доступу до агента), ідентифікатор об'єкта MIB, поле даних і тип даних, тип команди, статус помилки.

За допомогою функцій бібліотеки wsnmp32 створюється і надсилається у мережу таке SNMP-повідомлення:

```
Result = SnmpSetTranslateMode(SNMPAPI_TRANSLATED)
srcEntity = SnmpStrToEntity(hSession, txSource)
...
Result = SnmpSetTranslateMode(SNMPAPI_UNTRANSLATED_V1)
srcEntity = SnmpStrToEntity(hSession, txSource)
```

```
...
Result = SnmpSetTranslateMode(SNMPAPI_TRANSLATED)
dstEntity = SnmpStrToEntity(hSession, txDest)
...
Result = SnmpSetTranslateMode(SNMPAPI_UNTRANSLATED_V1)
dstEntity = SnmpStrToEntity(hSession, txDest)
...
NullValue.syntax = SNMP_SYNTAX_NULL
contextOct.len = Len(txContext.Text)
contextOct.ptr = txContext
Result = SnmpSetTranslateMode(SNMPAPI_TRANSLATED)
context = SnmpStrToContext(hSession, contextOct)
...
Result = SnmpStrToOid(txOid, getOid)
vbl = SnmpCreateVbl(hSession, getOid, NullValue)
pdu = SnmpCreatePdu(hSession, SNMP_PDU_GET,1234,0,0,vbl)
Result = SnmpSendMsg(hSession, srcEntity, dstEntity, context, pdu)
```

Функція SnmpStrToEntity перетворює поданий у текстовому форматі IP-адрес і контекст у формат SNMP-повідомлення. Так само працює функція SnmpStrToOid, яка перетворює ідентифікатор об'єкта MIB у формат SNMP-повідомлення. Функція SnmpCreateVbl створює зв'язний список даних, куди входять ідентифікатор об'єкта MIB, тип даних і самі дані. Функція Snmp CreatePdu створює та ініціює елемент даних протоколу SNMP. За допомогою функції SnmpSendMsg SNMP-повідомлення надсилається агенту системи моніторингу. В цій функції необхідно вказати IP-адресу менеджера та IP-адресу агента.

Для аналізу SNMP-повідомлень, одержаних від агента, використовується функція RecvMsg, в якій за допомогою SnmpRecvMsg отримують SNMP-повідомлення, надіслане агентом за допомогою SnmpSendMsg:

```
Result = SnmpRecvMsg(hSession, srcEntity, dstEntity, context, pdu)
Result = SnmpGetPduData(pdu, pdutype, ReqID, errStat, errInd, vbl)
Result = SnmpGetVb(vbl, 1, getOid, getValue)
```

Функція SnmpGetPduData повертає вибрані поля даних елементу даних протоколу SNMP. Функція SnmpGetVb вибирає інформацію зі змінного списку даних. Отриману інформацію з SNMP-повідомлення виводять у TextBox форми ManagerForm, перевіряючи також, від якого агента одержано інформацію.

Генерування SNMP-запитів відбувається за допомогою програмної компоненти Timer. Ця компонента через вказаний проміжок часу викликає відповідну процедуру обробки події, яка відтак викликає процедуру SendRequest, що формує і надсилає SNMP-запити агентам системи моніторингу. Процедурі txSnmpListen_KeyPress передаються отримані від агентів SNMP-повідомлення, а також повідомлення про втрачені SNMP-повідомлення. Процедура txSnmpListen_KeyPress для опрацювання отриманих SNMP-повідомлень викликає процедуру RecvMsg, описану вище.

Форма користувацького інтерфейсу OptionForm призначена для введення таких параметрів, як IP-адреса консолі управління системи моніторингу, пароль доступу до консолі управління (Context), ідентифікатор об'єкта MIB (OID Monitoring), максимально допустиме значення змінної, за якою ведеться постійне спостереження (Max value), назва ідентифікатора об'єкта MIB, проміжок часу, через який надсилають відповідні SNMP-запити до агентів системи моніторингу. Форма користувацького інтерфейсу GraphikForm відображає у вигляді графіка зміну значення об'єкта бази керуючої інформації MIB у часі. Форма користувацького інтерфейсу BrowserForm призначена для перегляду значень об'єктів MIB. Для цього використовують аналогічні процедури надсилання, отримання та обробки SNMP-повідомлень, які реалізовано у ManagerForm.

Висновки

Розглянуто проблему моніторингу комп'ютерної мережі за умов обмежень на використання системних ресурсів та високих вимог до надійності й живучості системи моніторингу. Розроблено автономну децентралізовану систему моніторингу комп'ютерної мережі, яка складається з колективу програмних агентів. Кожний агент може працювати в двох режимах: основному режимі та режимі консолі управління системою моніторингу. В основному режимі агент збирає інформацію про комп'ютерну мережу. В режимі консолі управління агент надає користувачу доступ до зібраної усіма агентами інформації та дозволяє виконувати команди управління системою моніторингу.

Розроблена система моніторингу дає змогу отримувати достовірнішу інформацію про роботу мережі з більшою оперативністю в умовах обмежень на використання обчислювальних та мережевих ресурсів, які задає користувач. Автономна система моніторингу побудована на основі концепції багатоагентних систем, у межах якої окремий програмний агент системи володіє певною ініціативою щодо планування та реалізації сценаріїв моніторингу. В роботі програмних агентів реалізовано методи організації адаптивних процесів збирання інформації з використанням принципів самоорганізації та концепції структурної адаптації. Це дає змогу автономній системі моніторингу отримувати достовірнішу та оперативнішу інформацію про роботу комп'ютерної мережі, завдяки чому якісно покращується процес управління мережею. Використання автономної системи моніторингу дає змогу забезпечити істотне зменшення кількості людино-годин, витрачених на підтримання мережі в робочому стані, за рахунок передавання частини повноважень із управління мережею програмним агентам системи моніторингу.

Запропоновано децентралізовану програмну архітектуру автономної системи моніторингу, в якій немає центру управління. За рахунок цього забезпечуються високі надійність та живучість системи моніторингу. В програмній архітектурі автономної системи моніторингу реалізовано прикладний програмний інтерфейс SMA та відповідну програмну бібліотеку, яка дає змогу збирати статистичні дані про роботу комп'ютерної мережі та її вузлів. Розглянуто реалізацію програмного агента та консолі управління автономної системи моніторингу комп'ютерної мережі.

Список літератури

1. George Varghese, Jun Xu (2022). Chapter 16 - Measuring network traffic, in Network Algorithmics, 2nd ed., George Varghese, Jun Xu (eds.), Morgan Kaufmann, 449–488. DOI: 10.1016/B978-0-12-809927-8.00024-5

2. P.-W. Tsai, C.-W. Tsai, C.-W. Hsu and C.-S. Yang (2018). Network Monitoring in Software-Defined Networking: A Review, IEEE Systems Journal, Vol. 12, No. 4, 3958–3969. DOI: 10.1109/JSYST.2018.2798060

3. A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia and P. Casas (2019). A Survey on Big Data for Network Traffic Monitoring and Analysis, IEEE Transactions on Network and Service Management, Vol. 16, No. 3, 800–813. DOI: 10.1109/TNSM.2019.2933358

4. Na Xia et. al. (2022). Optimization algorithms in wireless monitoring networks: A survey, Neurocomputing, Vol. 489, 584–598. DOI: 10.1016/j.neucom.2021.12.072

5. Lee, S., Levanti, K., & Kim, H. S. (2014). Network monitoring: Present and future. Computer Networks, Vol. 65, 84–98. DOI: 10.1016/j.comnet.2014.03.007

6. Shi, Peng & Yan, Bing. (2020). A Survey on Intelligent Control for Multiagent Systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 1–15. DOI: 10.1109/TSMC.2020.3042823.

7. Niu, Y., Miao, K., Liu, T., Wu, L. (2023). Survey on Coordination Problems of Multi-agent System and Application in Unmanned Systems. In: Proceedings of 2022 International Conference on Autonomous Unmanned Systems (ICAUS 2022). ICAUS 2022. Lecture Notes in Electrical Engineering, Vol. 1010. Springer, Singapore. DOI: 10.1007/978-981-99-0479-2_180

8. Dorri, A., Kanhere, S., Jurdak, R. (2018) Multi-Agent Systems: A Survey, in IEEE Access, Vol. 6, 28573–28593. DOI: 10.1109/ACCESS.2018.2831228.

9. Rizk, Y., Awad, M., Tunstel, E. (2018) Decision Making in Multi-Agent Systems: A Survey, in IEEE Transactions on Cognitive and Developmental Systems, Vol. 10, No. 3, 514–529. DOI: 10.1109/TCDS.2018.2840971.

10. Amirkhani, A., Barshooi, A.H. (2022) Consensus in multi-agent systems: a review, Artificial Intelligence Review, 55, 3897–3935. DOI: 10.1007/s10462-021-10097-x

11. Botchkaryov, A., Golembo, V., Paramud, Y., Yatsyuk, V. (2019). Cyber-physical systems: data collection technologies, A. Melnyk (ed.), Lviv: Magnolia 2006. 176 p. (in Ukrainian). ISBN: 98-617-574-139-9

12. Botchkaryov A. (2020). Structural adaptation of data collection processes in autonomous distributed systems using reinforcement learning methods, Computer Systems and Networks, Lviv Polytechics, Issue 2, No. 1, 13–26. (in Ukrainian). DOI: 10.23939/csn2020.01.013

13. Sharma, D. P., Singh, B. K., Gure, A. T., Choudhury, T. (2021). Autonomic Computing: Models, Applications, and Brokerage. In: Autonomic Computing in Cloud Resource Management in Industry 4.0., Choudhury, T. et al. (eds.), Springer, Cham, 59–90. DOI: 10.1007/978-3-030-71756-8_4

14. Dehraj, P., Sharma, A. (2021). A review on architecture and models for autonomic software systems, Journal of Supercomputing, 77, 388–417. DOI: 10.1007/s11227-020-03268-0

AUTONOMOUS DECENTRALIZED COMPUTER NETWORK MONITORING SYSTEM BASED ON SOFTWARE AGENTS

A. Botchkaryov

Lviv Polytechnic National University,
Computer Engineering Department

The problem of monitoring a computer network under conditions of limitations on the use of system resources and high requirements for the survivability of the monitoring system has been considered. An autonomous decentralized computer network monitoring system has been developed, consisting of a team of software agents. Each agent can operate in two modes: main mode and monitoring system management console mode. In the main mode, the agent collects information about the computer network. In management console mode, the agent provides the user with access to information collected by all agents and allows the user to execute commands to manage the monitoring system.

The developed monitoring system allows you to obtain more reliable information about the operation of the network with greater efficiency under the conditions of limitations on the use of system resources specified by the user. The autonomous monitoring system is created on the basis of the concept of multi-agent systems, within which a software agent of the system has some initiative for planning and implementing monitoring scenarios. The operation of software agents implements methods for organizing adaptive processes for collecting information using the principles of self-organization and the concept of structural adaptation.

A decentralized software architecture for an autonomous monitoring system without a control center has been proposed. This ensures high reliability and survivability of the monitoring system. The software architecture of the autonomous monitoring system implements the SMA application software interface and the corresponding software library, which allows you to collect statistical data on the operation of the computer network and its nodes. The implementation of a software agent and a management console for an autonomous computer network monitoring system has been considered.

Key words: computer network monitoring; autonomous system; decentralized control; software agent.