ISSN 2225-5036 (Print), ISSN 2411-071X (Online)
http://infosecurity.nau.edu.ua; http://jrnl.nau.edu.ua/index.php/Infosecurity

ЗАХИСТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ОБЛАДНАННЯ/ SOFTWARE & HARDWARE ARCHITECTURE SECURITY

DOI: 10.18372/2225-5036.30.18578

ПОБУДОВА ПЕРЕВІРНОЇ ТЕСТОВОЇ ПОСЛІДОВНОСТІ ДЛЯ ОЦІНКИ ТЕХНІЧНОГО СТАНУ ОБ'ЄКТІВ З ВБУДОВАНИМ ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

Василь Кузавков, В'ячеслав Солодовник, Юлія Болотюк

Військовий інститут телекомунікацій та інформатизації імені Героїв Крут

КУЗАВКОВ Василь Вікторович, д.т.н., професор.
Рік та місце народження: 1966 рік, м. Дзержинськ, Мінська область, Білорусь.
Освіта: Київське вище інженерне радіотехнічне училище протиповітряної оборони імені Маршала авіації О. І. Покришкіна, 1988 рік.
Посада: начальник кафедри побудови телекомунікаційних систем з 2016 року.
Наукові інтереси: методи і засоби технічного діагностування, теорія автоматичного та автоматизованого управління, електромагнітна сумісність, технічні канали витоку інформації.
Публікації: понад 150 наукових публікацій, серед яких наукові статті, монографії, навчальні посібники, тези та матеріали доповідей на конференціях.
E-mail: nevse@ukr.net.
Orcid ID: 0000-0002-0655-9759.

СОЛОДОВНИК В'ячеслав Ігорович, к.т.н.
Рік та місце народження: 1987 рік, м. Прага, Чехія.
Освіта: Військовий інститут телекомунікацій та інформатизації Національний технічний університет України «Київський політехнічний інститут», 2010 рік.
Посада: доцент кафедри телекомунікаційних систем та мереж.
Наукові інтереси: високошвидкісна завадостійка передача інформації у каналах телекомунікаційних систем, математичне моделювання.
Публікації: більше 30 наукових публікацій, серед яких наукові статті, монографії, навчальні посібники, патенти, тези та матеріали доповідей на конференціях.
E-mail: viacheslav.solodovnyk@viti.edu.ua.
Orcid ID: 0000-0002-9113-7672.

БОЛОТЮК Юлія Володимирівна, ад'юнкт науково організаційного відділу військового інституту телекомунікацій та інформатизації імені Героїв Крут.
Рік та місце народження: 1993 рік, м. Прилуки, Чернігівська область, Україна.
Освіта: Військовий інститут телекомунікацій та інформатизації Державного університету телекомунікацій, 2015 рік.
Наукові інтереси: методи і засоби технічного діагностування.
Публікації: 10 наукових публікацій, серед яких наукові статті, патенти, тези та матеріали доповідей на конференціях.
E-mail: yuliia.bolotiuk@viti.edu.ua.
Orcid ID: 0000-0002-3805-6419.

Анотація. *Сучасні засоби обміну інформації в своїй більшості побудовані як системи з вбудованим програмним забезпеченням. Процес функціонування таких систем полягає у взаємодії програмних та апаратних складових. Архітектура таких систем відповідає моделі взаємодії відкритих систем (Open System Interconnection Basic Reference Model, OSI), яка визначає певні рівні у мережах, дає їм стандартні імена та вказує, які функції має виконувати кожен рівень. Порушення функціонування подібних систем можуть бути спричинені випадковими відмовами та дефектами апаратної складової, помилками та відмовами програмного забезпечення, або як наслідок зовнішнього впливу. Складність оцінки технічного стану (технічного діагностування) обумовлена значним територіальним рознесенням елементів системи, різноманіттям функціональних вузлів та специфікацій устаткування, а також використанням в системах устаткування подвійного призначення.*

Враховуючи складність обраного об'єкту, завдання автоматизації контролю покладається на комп'ютерно-вимірювальні системи. Аналіз архітектури обраного об'єкту контролю (наявність формалізованих – стандартизованих рівнів) дозволів запропонувати розв'язання завдань технічного діагностування (або завдань контролю фізичної цілісності) шляхом поетапного (покрокового) тестування окремих рівнів апаратно-технічних засобів. Представлена методика перевірки спирається на організаційні заходи (наявність щоденного контролю функціонування) та використання стандартизованих наборів протоколів кожного рівня моделі OSI. При цьому, на кожному етапі перевірки відбувається реєстрація двох складових діагностичного параметру (часової та енергетичної). Наявність математичних моделей процесів старіння апаратної частини та фізичного середовища передачі даних дозволило отримати еталонні значення для діагностичного параметру при будь якому часі функціонування об'єкту контролю, або визначити зміну основних параметрів при моделюванні поведінки системи на визначений час. Сукупність чисельних значень обох складових діагностичного параметру дозволяє визначити технічний стан не лише апаратної частини об'єкту контролю і правильність функціонування програмної частини на різних рівнях моделі OSI, но і наявність (або відсутність) в системі несанкціонованого впливу (програмного або апаратного).

***Ключові слова**: технічний стан, діагностична інформація, фізична цілісність, комунікаційне обладнання, вбудоване програмне забезпечення, модель, математичне моделювання.*

Постановка проблеми

Однією із основних задач, які виникають при експлуатації систем з вбудованим програмним забезпеченням (ВПЗ) є вчасне виявлення потенційних проблем (отримання інформації, щодо ймовірностей виникнення помилок або зовнішнього втручання) під час експлуатації. Порушення у функціонуванні апаратної або програмної складової можуть спричиняти непередбачений стан чи поведінку системи та приводити до значних матеріальних збитків.

Задача визначення технічного стану (контролю фізичної цілісності) особливо актуальна для комплексних територіально рознесених технічних систем, які складаються з багатьох підсистем різноманітного призначення і складності. Загальною рисою подібних систем є використання обчислювальних засобів та складових з ВПЗ.

Наведений в роботі підхід до побудови перевірної тестової послідовності призначений для отримання часової складової енерго-часового діагностичного параметру об'єкта контролю (ОК).

Аналіз основних досліджень і публікацій

Для об'єктів контролю, які потрапляють під визначення "засоби з ВПЗ" визначення стану є одним з найбільш перспективних завдань, яке дозволяє не лише передбачити момент виходу об'єкта (чи його частини) з працездатного стану і визначити періодичність проведення операцій технічного обслуговування, а також використати отримані результати для створення алгоритмів контролю технічного стану [1, 2].

Враховуючи розмірність об'єкта контролю (як просторову так і кількісну) рішення завдань технічної діагностики неможливе без використання комп'ютерно-вимірювальних систем з елементами підтримки прийняття рішення. Дані, отримані під час контролю дозволять підвищити ефективності системи технічного обслуговування, забезпечити заданий рівень показників технічного забезпечення, значної економії матеріальних та фінансових коштів [3, 4].

Мета та постановка завдання

Задача полягає у розробці нових методів отримання та обробки діагностичної інформації для визначення технічного стану радіоелектронного засобів з ВПЗ що входять до складу комунікаційного обладнання.

Результати також використовуватимуться для вирішення ще одного завдання технічної діагностики – пошуку місця відмови (відповідного рівня згідно моделі TCP/IP).

Виклад основного матеріалу дослідження

Формалізація опису будь якої технічної системи є одним з основних питань технічного діагностування. З позицій системного підходу, обраний ОК це сукупність взаємодіючих однотипних елементів обчислювальної техніки та програмного середовища яка забезпечує виконання алгоритмів обробки інформації, генерування, споживання, транспортування інформаційних потоків. Крім того, для опису процесів в обраному ОК можливо використовувати відому модель OSI.

Слід розуміти, що модель OSI абстрактна і використовується лише як еталон для вивчення і розуміння комп'ютерних мереж. Не всі рівні моделі однаково актуальні в сценаріях мережевого зв'язку. Крім того, мережі часто використовують комбінацію різних протоколів і технологій для надання послуг і з'єднання [5]. Тому, користувачі користуються дещо іншим способом опису обраного ОК, а саме моделлю TCP/IP.

Запропонований підхід визначення фактичного технічного стану обраного ОК призначений для забезпечення функціонування прийнятої системи технічного обслуговування (планово-попереджувальної системи), оскільки для зразків подвійного призначення виконання процедури щоденного контролю функціонування не передбачено виробником.

В роботі запропоновано варіант програмної складової для здійснення перевірки. Розроблена структура перевірної тестової послідовності дозволяє отримати значення часової складової діагностичного параметру при покроковій перевірці технічного стану кожного з рівнів уніфікованої моделі, за якою збудовано ОК. Отже, перевірна послідовність зорієнтована на формування запиту та отримання відповіді від певного рівня ОК (системи з ВПЗ побудованої за моделлю OSI).

Інформаційний відгук (діагностичний параметр (ДП)) на перевірну послідовність має чіткі еталонні енергетичні та часові значення для кожного з рівнів моделі. Представлено схему проходження перевірної тестової послідовності (рис. 1) [6].

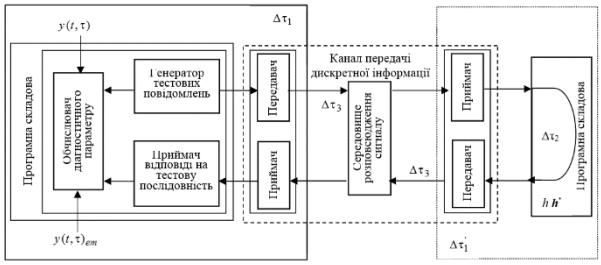

Рис. 1. Схема проходження тестових послідовностей в ОК з вбудованим програмним забезпеченням

На рисунку 1 позначено: $y(t,\tau)$, $y(t,\tau)_{ет}$ – ДП та еталонне значення ДП, які залежать від конструктивних параметрів: $\dot{h}$ – апаратної складової ОК, амплітудної та фазочастотної характеристики тракту формування та передачі тестових повідомлень; t – час напрацювання ОК; τ – тривалості перевірних тестових послідовностей; $\Delta\tau_{\Sigma} = \Delta\tau_1 + \Delta\tau_2 + \Delta\dot{\tau}_1 + \Delta\tau_3$ – сумарне приростання часу; $\Delta\tau_1\Delta\dot{\tau}_1$ – часове приростання, пов'язані з функціонуванням апаратної частини ОК; $\Delta\tau_2$ – приростання часу, яке характеризує затримки, обумовлені дисципліною обслуговування (обробки) тестових повідомлень у термінах теорії масового обслуговування, вплив параметрів налаштування операційного середовища h; $\Delta\tau_3$ – приріст часу який характеризує «старіння» середовище передачі даних (і наслідок цього старіння - зміну смуги пропускання середовища передачі даних). Слід відзначити, що складова $\Delta\tau_2$ також відображує в собі параметри апаратної частини $\dot{h}$ (обсяг пам'яті, частота процесору, частота опитування пристроїв вводу-виводу і т. ін.), оскільки обчислювальні процеси виконуються відповідною складовою апаратної частини.

Чисельні значення енергетичної складової ДП $y(t,\tau)$ – визначається змінами струму, які реєструються безконтактним методом на шині живлення об'єкту який формує тестову послідовність і отримує відповідь на неї під час контролю функціонування [7]. Ці значення залежить не тільки від конструктивних параметрів h і перевірних тестів тривалістю τ, а і від часу напрацювання об'єкту контролю t, тобто відображують процес «старіння» апаратної частини.

Зміна параметрів апаратної частини в часі (старіння) обумовлено фізико-хімічними процесами як в напівпровідникових структурах та компонентах об'єкту контролю так і середовища передачі даних.

Оскільки модель OSI є абстрактна і не має фізичного представлення всіх рівнів, то для визначення технічного стану обраного ОК запропоновано проводити перевірку функціонування ОК на трьох рівнях моделі TCP/IP (прикладний, транспортний та мережевий). Співставлення рівнів моделі OSI та TCP/IP розглянуто в [8]. Процес (методику) перевірки ОК з застосуванням тестової послідовності розглянемо на прикладі фрагменту комунікаційній мережі (рис. 2). Наведений фрагмент ОК складається з територіально рознесених компонентів системи різного ступеня апаратної та програмної складності.

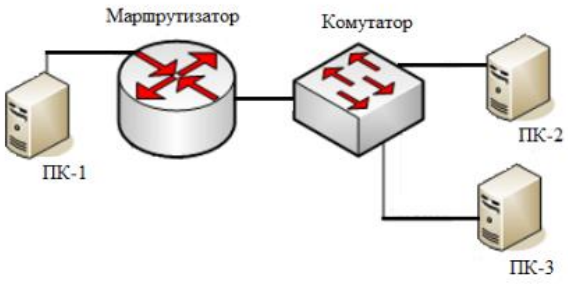

Рис. 2. Приклад сегменту комунікаційної мережі

До складу фрагменту мережі входять наступні елементи:

- маршрутизатор - мережевий пристрій, який на підставі інформації про топологію мережі та певних правил приймає рішення про пересилання пакетів мережевого рівня (рівень 3 моделі OSI) між різними сегментами мережі;
- комутатор - пристрій, призначений для з'єднання декількох вузлів комп'ютерної мережі в межах одного сегмента. В залежності від специфікації та технічних характеристик, комутатори можуть працювати на різних рівнях моделі OSI. Комутатор рівня L2 - працює на канальному рівні (рівень 2 моделі OSI), і тому в загальному випадку може тільки поєднувати вузли однієї мережі по їхніх MAC-адресах. Комутатор рівня L3 - працюють на мережевому рівня (рівень 3 моделі OSI), відрізняють IP-адреси пристроїв, визначають найкоротші маршрути. Комутатор рівня L4 – відповідають за забезпечення надійності передачі даних, на підставі інформації з заголовків пакетів відрізняють приналежність трафіку різних додатків здатні перенаправляти трафік на підставі вхідної інформації;
- персональний комп'ютер (ПК) – обчислювальний пристрій з вбудованим програмним забезпеченням.

Наведена спрощена блок-схема алгоритму перевірки (визначення фактичного технічного стану) фрагменту телекомунікаційної мережі (рис. 3).

Процедура контролю починається з визначення всіх IP адрес об'єктів, які входять до сегменту мережи (див. рис. 2).

Почергово, за заздалегідь визначеним графіком, під час щоденного контролю функціонування один з об'єктів мережі розпочинає відправку тестових повідомлень (на певний час виконує функцію Клієнт мережі).

Адресацію перевірних тестових повідомлень складено таким чином, щоб відповідач (тимчасово - Сервер мережи) мав можливість сформувати відповідь на кожному з визначених рівнів моделі, починаючи з мережевого.

У зв'язку з відсутністю системи єдиного часу, часові інтервали відправки перевірного повідомлення та отримання відповіді реєструються відправником. Енергетична складова діагностичного параметра саме в ці інтервали часу проходження тестових повідомлень також фіксується відправником із застосуванням датчика діагностичної інформації безконтактним індукційним методом. Обидві складові діагностичного параметру забезпечують функціонування системи прийняття рішень.

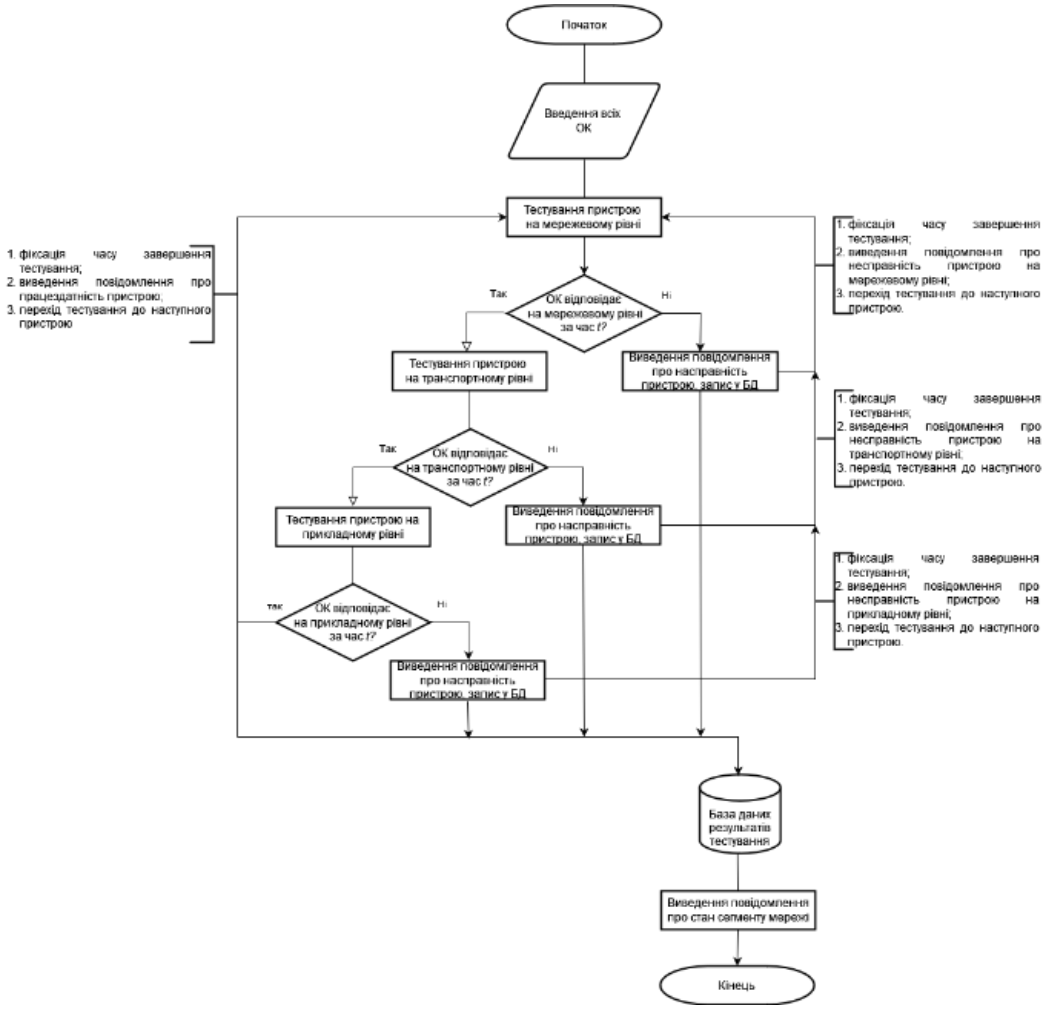

Рис. 3 Блок-схема алгоритму діагностування

В сучасних комунікаційних мережах, керування сеансами здійснюється на вищих рівнях або в протоколах прикладного рівня. Як наслідок, не існує чітко визначених методів для взаємодії різних компонентів прикладного програмного інтерфейсу (application programming interface, API) або бібліотеки для спеціальної перевірки рівня сеансу. Проте, керуванням і завершенням сеансу у своїй програмі можливо непрямо, а за допомогою бібліотек, фреймворків або протоколів. Фрагмент програмного коду ініціалізації об'єктів сегменту мережі та формування повідомлення про їх стан (блок 1), написаний на мові програмування Java, наведено нижче.

блок 1

```
    String hostName = "x.x.x.x"; // присвоєння IP адреса ОК
    try {
      InetAddress host = InetAddress.getByName(hostName);
      String canonicalHostName = host.getCanonicalHostName(); // ім'я ОК
      boolean reachable = host.isReachable(nnnn); // nnnn-час очікування відповіді

      if (reachable) {
        System.out.println(hostName + " " + canonicalHostName + "доступний.");
      } else {
        System.out.println(hostName + " " + canonicalHostName + "не доступний.");
      }
    } catch (IOException e) {
      System.err.println("Error: " + e.getMessage());
    }
```

У цьому прикладі: x.x.x.x позначено IP адресу ОК на який відправлено тестове повідомлення; nnnn – час очікування на відповідь. Якщо, час очікування відповіді перевищує встановлене значення то на цьому етапі перевірки стан ОК визнається як несправний. У іншому випадку, ОК визнається як справний і згідно блок-схеми алгоритму (див. рис. 3) формується тестова послідовність для почергової перевірки рівнів моделі TCP/IP (міжмережевого, транспортного, прикладного).

Для операцій з об'єктами (вузлами) мережі використовується клас InetAddress бібліотеки java.net, який повною мірою характеризує атрибути і поведінку ОК.

Змінна hostName використана для визначення IP адреси відділеного ПЕОМ, який підлягає діагностуванню.

Функцію InetAddress.getByName(hostName) використано для програмного відображення фізичного об'єкту контролю.

Функцію isReachable (nnnn) застосовано для перевірки досяжності об'єкту контролю у межах визначеного раніше інтервалу часу.

У якості прикладу формування тестової послідовності перевірки прикладного рівня розроблено

клієнт-серверний застосунок. Тому нижче, наведено фрагмент лістингу клієнтської частини клієнт-серверного застосунку для прикладного рівня моделі TCP/IP (блок2).

блок 2

```
public class DNSClient {
  public static void main(String[] args) {
    String serverAddress = "localhost";
    int serverPort = №XX;

    try (Socket socket = new
Socket(serverAddress, serverPort);
         BufferedReader in = new
BufferedReader(new
InputStreamReader(socket.getInputStream()));
         PrintWriter out = new
PrintWriter(socket.getOutputStream(), true)) {

//Надсилання повідомлення на сервер (ОК)
         out.println("ти в наявності?");

//Отримання та виведення відповіді з серверу
(ОК)
         String serverResponse =
in.readLine();
         System.out.println("я присутній: "
+ serverResponse);

    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

З точки зору роботи підсистеми діагностування сформований запит та отримана відповідь є перевірною тестовою послідовністю для якої фіксується час відправлення запиту та час отримання відповіді (часова складова енерго-часового діагностичного параметру).

Клас Public class DNSClient, маючи у своєму розпорядженні, IP-адресу сервера (ОК) та номер порту, забезпечує підключення клієнта до DNS-сервера на ОК.

Об'єкт класу Socket встановлює з'єднання з сервером за вказаною адресою та портом.

Об'єкти класів BufferedReader і PrintWriter призначені для читання символів з введеного потоку та виведення текстових даних у вихідний потік.

Команда out.println відправляє повідомлення на сервер (ОК), а процедура System.out.println відповідає за отримання та виведення відповіді з серверу (ОК).

Розглянемо фрагмент програмної складової (лістинг) серверної частини клієнт-серверного застосунку для прикладного рівня моделі TCP/IP (блок 3).

блок 3

```
public class DNSServer {
  public static void main(String[] args) {
    int port = №XX;

    try (ServerSocket serverSocket = new
ServerSocket(port)) {
         System.out.println("DNS-сервер про-
слуховує порт" + port);

         while (true) {
         try (Socket clientSocket =
serverSocket.accept();
              BufferedReader in = new
BufferedReader(new
InputStreamReader(clientSocket.getInputStream()
));
              PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(),
true)) {

    // Читання повідомлення від клієнта

              String clientMessage =
in.readLine();
              System.out.println("отримано
повідомлення від клієнта: " + clientMessage);

    // Перевірка, чи клієнт отримав відповідь
на запит
              if ("ти в наявно-
сті?".equalsIgnoreCase(clientMessage)) {
    // Виведення статусу сервера
              out.println("DNS-сервер в наяв-
ності і працездатний!");
              } else {
                  //  Повідомлення про поми-
лку
                  out.println("Помилкове по-
відомлення");
                  }
               }
              }
         } catch (IOException e) {
         e.printStackTrace();
    }
  }
}
```

Блок 3 це програмна складова перевірки функціонування серверної частини клієнт-серверного застосунку. Команди, процедури та функції подібні тим, що описано в блоці 2.

Об'єкт класу clientSocket приймає з'єднання від клієнта за допомогою serverSocket.accept(), після чого використовує цей clientSocket для взаємодії з клієнтом.

Команда out.println відправляє повідомлення клієнту, а процедура System.out.println відповідає за отримання та виведення відповіді клієнтом.

Правильність функціонування наведеного вище клієнтського та серверного застосунку може бути перевірена спеціальним додатком, який складається з клієнтської та серверної частин.

блок 4.1(клієнт)

```
Unit test for server and client above  //
Тестова послідовність для сервера та клієнта)

public class SimpleDNSClientTest {

  @Test
  public void testDNSClient() {
    ByteArrayOutputStream outputStream =
new ByteArrayOutputStream();
     System.setOut(new
PrintStream(outputStream));

     SimpleDNSClient.main(new String[]{});

     String consoleOutput =
outputStream.toString();
  //перевірка працездатності клієнт-сервер-
ного застосунку
     assertEquals("DNS-запит надіслано: зра-
зок DNS-запиту\nDNS-відповідь отримано: зразок
DNS-відповіді\n ", consoleOutput);

     System.setOut(System.out);
     }
}
```

блок 4.2(сервер)

```
public class SimpleDNSServerTest {

  @Test
  public void testDNSServer() {
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    System.setOut(new PrintStream(outputStream));

    SimpleDNSServer.main(new String[]{});

    String consoleOutput = outputStream.toString();
    assertEquals("DNS-сервер запущено на порту №XX\n", consoleOutput);

    System.setOut(System.out);
  }
}
```

В наведеному прикладі клас SimpleDNSClientTest використано для написання юніт-тестів, які перевіряють ізольовані частини програмного коду клієнтського застосунку.

Аннотация @Test використовується для позначення того, що ця функція повинна бути викликана під час виконання тестів.

Потік виведення ByteArrayOutputStream використовується для перехоплення виводів, які надсилається операторам.

Функція System.setOut(new PrintStream (outputStream)) - замінює системний потік стандартного виведення на потік ByteArrayOutputStream, щоб мати змогу перехопити вивід, який надсилається на сервер

Функція SimpleDNSClient.main(new String[]{}) – викликає функцію main класу SimpleDNSClient з порожнім масивом у якості аргументів.

Функція String consoleOutput = outputStream. toString() – призначена для відображення на консолі повідомлення під час виконання функції main.

Функція AssertEquals – призначення для порівняння змінних, порівнює перехоплені потоки даних і у випадку збігу виводить відповідне повідомлення на екран.

У випадку коректного функціонування застосунку отримаємо повідомлення: «DNS-запит надіслано: зразок DNS-запиту \n DNS-відповідь отримано: зразок DNS-відповіді \n», де в змінну \n передається зразок запиту\відповіді.

Функція System.setOut(System.out) - повертає системний потік стандартного виведення до його звичайного стану для того, щоб не впливати на інші тести чи ділянки коду. Цей тест перевіряє, чи клас SimpleDNSClient отримує очікувану відповідь від ОК під час своєї роботи. Тест вважається пройденим, якщо оператор отримав відповідне повідомлення.

Для блоку 4.2, клас SimpleDNSServerTest, функція AssertEquals - призначена для порівняння змінних, порівнює перехоплені потоки даних і у випадку збігу виводить на екран повідомлення з номером порту, який використано під час перевірки.

Функція System.setOut(System.out) - повертає системний потік стандартного виведення до його звичайного стану для того, щоб не впливати на інші тести чи ділянки коду. Цей тест перевіряє, чи клас SimpleDNSServer дійсно відправляє повідомлення (відповідь) про стан ОК. Якщо вивід відповідає очікуваному, то тест вважатиметься успішним. Наведені вище фрагменти блок-схем та програмних кодів дозволяють реалізувати на практиці процес перевірки технічного стану обраного ОК на одному з рівнів моделі із застосуванням двофакторного (енерго-часового) діагностичного параметру.

Наявність часу, вільного від виконання функціональних завдань, забезпечується прийнятою і реалізованою на сьогодні системою технічного обслуговування.

Автоматизацію процесу перевірки усього територіально рознесеного ОК забезпечить впровадження «планувальника завдань», приклад якого наведено нижче.

```
public class ScheduledTaskPlanner {
  public static void main(String[] args) {
    ScheduledExecutorService executorService = Executors.newScheduledThreadPool(2);

// поточний час
    LocalTime currentTime = LocalTime.now();

  // розрахунок інтервалу до часу t наступного дня
    long initialDelay = calculateInitialDelay(currentTime);

// запуск DNS-сервера щодня в час t з таймаутом N мілісекунд
    ScheduledFuture<?> serverFuture = executorService.scheduleWithFixedDelay(ScheduledTaskPlanner::runDNSServer,
        initialDelay, 24, TimeUnit.HOURS);

// запуск DNS-клієнта щодня в час t з таймаутом N мілісекунд
    ScheduledFuture<?> clientFuture = executorService.scheduleWithFixedDelay(ScheduledTaskPlanner::runDNSClient,
        initialDelay, 24, TimeUnit.HOURS);

// встановлено час очікування для кожного завдання (N мілісекунд)
    long timeout = N; // milliseconds

    scheduleTimeout(executorService, serverFuture, timeout);
    scheduleTimeout(executorService, clientFuture, timeout);
  }

  private static void runDNSServer() {
    System.out.println("Running DNS Server at t am...");
  //  SimpleDNSServer.main(новий рядок[]{});
// встановлення параметрів існуючого DNS-сервера

  }

  private static void runDNSClient() {
    System.out.println("Running DNS Client at t am...");
  // Replace with your DNS client logic
  // SimpleDNSClient.main(new String[]{});
  //  SimpleDNSClient.main(новий рядок[]{});
//  встановлення параметрів існуючого DNS-клієнта

  }

  private static long calculateInitialDelay(LocalTime currentTime) {
```

```
    // Планування запуску діагностичного тесту на час t кожного дня
        LocalTime targetTime = LocalTime.of(t, 0);
        if (currentTime.isAfter(targetTime)) {
        targetTime = targetTime.plusHours(24);
        }
        return java.time.Duration.between(currentTime, targetTime).toMillis();
      }

      private static void scheduleTimeout(ScheduledExecutorService executorService, ScheduledFuture<?> taskFuture, long timeoutMilliseconds) {
    //якщо час виконання діагностичного тесту буде відрізнятись від еталонного, то виводиться повідомлення про невідповідність на конкретному рівні моделі
        executorService.schedule(() -> {
        if (!taskFuture.isDone()) {
            System.out.println("Час діагностування відрізняється від еталонного. Тест припинено!");
            taskFuture.cancel(true);
            }
        }, timeoutMilliseconds, TimeUnit.MILLISECONDS);
      }
    }
```

Клас ScheduledTaskPlanner є прикладом використання ScheduledExecutorService для планування та виконання завдань визначених функціями runDNSServer та runDNSClient. Створюється процес ScheduledExecutorService за допомогою Executors. newScheduledThreadPool(2), що дозволяє планувати час проведення переевірки.

Параметр LocalTime.now() визначає поточний час. Викликається calculateInitialDelay(currentTime) для визначення відстані між поточним часом і наступним в час t. Після чого відбувається запуск планування виконання DNS сервера та DNS клієнта. Викликається scheduleWithFixedDelay для планування виконання завдань визначених у вищезгаданих функціях щодня в час t (відповідно 24 години), з початковою затримкою, розрахованою у попередньому кроці. Для встановлення тайм-ауту для кожного завдання викликається scheduleTimeout. Якщо завдання не виконане за визначений час, виводиться відповідне повідомлення, і завдання скасовується. Функції runDNSServer та runDNSClient представляють собою тестові дії, які виводять повідомлення про запуск DNS сервера та DNS клієнта. Даний фрагмент лістингу використовується як планувальник, який розпочинає виконання завдань щодня в час *t* і слідкує за їх виконанням, встановлюючи тайм-аут для кожного з них.

Висновки. Запропонована в роботі методика побудови перевірної тестової послідовності дозволяє визначити технічний стан обраного об'єкту контролю або може бути використана для перевірки фізичної цілісності даного об'єкту. Рішення приймається на підставі порівняння еталонних значень енерго-часового діагностичного параметру (значень отриманих при справному не підданому зовнішньому впливу об'єкті контролю) зі значеннями отриманими під час чергової перевірки функціонування (в межах прийнятої на озброєння системи технічного обслуговування). Методологічна підтримка прийнятого рішення здійснюється розробленими діагностичними моделями об'єкта контролю з урахуванням фізико-хімічних процесів старіння напівпровідникових структур апаратної частини та середовища передачі даних.

В наведених прикладах програмних застосунків відсутня складова реєстрації енергетичної складової діагностичного параметру. Реєстрація цієї складової здійснюється в проміжок часу між початком надсилання перевірної послідовності та часом отримання відповіді.

Список літератури

[1]. І.Ю. Субач, П.В. Хусаінов, В. А. Міщенко, Д.Е. Прусов. Структура системи підтримки прийняття рішень чергового адміністратора інформаційної мережі. Вісник Національного авіаційного університету. К.:НАУ, 2009. Том 3, № 40. С. 195-199.

[2]. О. В. Соломенцев, М. Ю. Заліський, О.А. Щербина, І. М. Яшанов. Методика визначення статистичних характеристик ефективності діагностування телекомунікаційних та радіоелектронних систем. Наукоємні технології. Київ: НАУ, 2021. № 4(52). С. 357-364. DOI: 10.18372/2310-5461.52.16381.

[3]. П.А. Шкуліпа, М.К. Жердєв, С.В. Лєнков, Ю.О. Гунченко. Шляхи і методи підвищення ефективності автономних автоматизованих систем технічного діагностування радіоелектронних пристроїв спеціального призначення. Сучасна спеціальна техніка. Київ. 2012. №3(30). С. 69-74.

[4]. Л.М. Сакович, С.І. Глухов, О.С. Бабій, А.О. Гальоса. Методика фізичного діагностування цифрових пристроїв об'єктів радіоелектронної техніки. Теоретичні основи розробки та експлуатації систем озброєння. Харків: ХНУПС, 2020. №2(62). С. 93-101. DOI: 10.30748/soivt.2020.62.12.

[5]. M. E. Ilchenko, A. V Moshinskaya, L. A. Urywsky. Levels separation and merging in the OSI reference model for information–telecommunication systems. Cybernetics and Systems Analysis. 2011. Vol. 47, No. 4, pp. 98-605. DOI: 10.1007/s10559-011-9340-4.

[6]. В. В. Кузавков, П. В. Хусаінов Прогнозування технічного стану однотипних програмно-апаратних засобів. Інформатика та математичні методи в моделюванні. Одеса: Національний університет «Одеська політехніка», 2018. № 1. С. 57-68.

[7]. В.В. Кузавков, О.Г. Янковський, Ю.В. Болотюк. Обгрунтування вибору показників оцінки ефективності функціонування автоматизованої системи контролю. Сучасні інформаційні технології у сфері безпеки та оборони. К.:НУОУ. 2022. Том 44, №2. С. 21-27. DOI: https://doi.org/10.33099/2311-7249/2022-44-2-21-27.

[8]. П.В. Хусаінов, І.Ю. Субач, О.В. Сілко, С. В. Любарський Основи побудови операційних систем, комплексів та засобів автоматизації управління військами: навчальний посібник. Київ: ВІТІ, 2016. 220 с.

УДК 621.317

Kuzavkov V., Solodovnyk V., Bolotiuk Y. Construction of a verifiable test sequence for assessing the technical condition of objects with embedded software

Abstract. *Modern means of information exchange are mostly built as systems with. The process of functioning of such systems consists in the interaction of software and hardware components. The architecture of such systems corresponds to the Open System Interconnection Basic Reference Model (OSI), which defines certain levels in networks, gives them standard names and indicates what functions each level should perform. Violations of the functioning of such systems can be caused by random failures and defects of the hardware component, errors and failures of the software, or as a result of external influence. The complexity of assessing the technical condition (technical diagnosis) is due to the significant territorial diversity of the system elements, the variety of functional nodes and specifications of the equipment, as well as the use of dual-purpose equipment in the systems. Given the complexity of the selected object, the task of automating control is entrusted to computer-measurement systems. Analysis of the architecture of the selected object of control (availability of formalized - standardized levels) allows to offer solutions to technical diagnostics tasks (or physical integrity control tasks) through step-by-step (step-by-step) testing of individual levels of hardware and technical means. The presented verification technique is based on organizational measures (availability of daily monitoring of functioning) and the use of standardized sets of protocols of each level of the OSI model. At the same time, at each stage of the check, two components of the diagnostic parameter (time and energy) are registered. The availability of mathematical models of the aging processes of the hardware part and the physical data transmission environment made it possible to obtain reference values for the diagnostic parameter at any time of the operation of the control object, or to determine the change of the main parameters when modeling the behavior of the system for a certain time. The set of numerical values of both components of the diagnostic parameter allows to determine the technical condition of not only the hardware part of the control object and the correct functioning of the software part at different levels of the OSI model, but also the presence (or absence) of unauthorized influence (software or hardware) in the system.*

Keywords: *technical condition, diagnostic information, physical integrity, communication equipment, embedded software, model, mathematical modeling.*

Кузавков Василь Вікторович, доктор технічних наук, доцент, начальник кафедри побудови телекомунікаційних систем Військового інституту телекомунікацій та інформатизації імені Героїв Крут.

Vasyl Kuzavkov, Dc.S, Head of the Department of Construction of telecommunication systems of the Military Institute of Telecommunications and Informatization named after the Heroes of Kruty.

Солодовник В'ячеслав Ігорович, кандидат технічних наук, доцент кафедри телекомунікаційних систем та мереж Військового інституту телекомунікацій та інформатизації імені Героїв Крут.

Viacheslav Solodovnyk, candidate of technical sciences, associate professor department of telecommunication systems and networks of the Military Institute of Telecommunications and Informatization named after the Heroes of Kruty.

Болотюк Юлія Володимирівна, ад'юнкт науково-організаційного відділу Військового інституту телекомунікацій та інформатизації імені героїв Крут.

Yuliia Bolotiuk, Adjunct (postgraduate) of the Scientific and organizational department of the Military Institute of Telecommunications and Informatization named after the Heroes of Kruty.

Отримано *26 січня 2024 року,* затверджено редколегією *1 квітня 2024 року*