UDK: 004.77, 004.728

HER VLADYSLAV
TARANIUK V.
TKACHENKO V.
KLYMENKO I.
NIKOLSKYI S.

METHODOLOGY OF NETWORK ENVIRONMENT TESTING FOR IoT DEVICES

Annotation: The article describes the basics of testing: writing test documentation (an example based on a report defect was proposed) and some testing methods. A performance test was also developed to test the load. Received basic knowledge of testing theory, as well as skills of writing and using bash scripts for performance tests.

Keywords: IoT, embedded system, test case, defect report, troubleshooting, performance testing.

1. Introduction. Importance of IoT Testing

1.1 What is the Internet of Things and IoT Testing?

The Internet of Things, or IoT, is a term for billions of devices connected to the Internet. Each of these devices collects data and exchanges it with other devices. The Internet of Things phenomenon is due to the presence of powerful network bandwidth, the proliferation of wireless networks and low-cost computer chips.

The Internet of Things connects various objects, adds modern sensors and facilitates data exchange between different devices in real time. This allows people to add a new level of digital intelligence to their devices and help them process information locally without any delay.

However, traditional software testing does not work for the Internet of Things. Testers should pay attention to user-centric testing and prevent errors, not detect them. This means that quality assurance engineers have a role to play both in operation and in development.

To guarantee the quality of the software, testers need to gain in-depth knowledge in the subject area. Testers who have no experience testing embedded systems or equipment should develop their skills in these areas.

1.2 Importance of IoT Testing

IoT is data exchange and collaboration in real time. Performance issues in any part can negatively affect the performance of another network. One node compromised as a result of a cyber attack can harm others.

Reliable Internet of Things testing ensures system predictability and prevents unexpected failures. You can easily identify weak network nodes in advance and take appropriate measures to increase system reliability. This will provide the end user with a flawless customer experience.

Vulnerabilities in IoT devices will help you determine the reliability of data privacy and digital security. In the absence of reliable cybersecurity, hackers can change the process or falsify data by gaining control of the IoT network. Therefore, before each update, you must thoroughly test the IoT device.

In addition, the fragmented nature of the Internet of Things ecosystem makes testing even more difficult. To do this, you need to have large and reliable test teams equipped with multiple platforms and devices to ensure greater compatibility between different channels.

In addition, to ensure that IoT applications work as expected, several other factors need to be considered:

- Ensure that your IoT devices are securely connected to sensors, the cloud, other IoT devices, and other elements needed to ensure unified interaction.
- Make sure the Internet of Things is able to ensure continuity

- Must meet international standards
- Harmonious and flawless work with other interconnected devices of the Internet of Things
- Data obtained from the Internet of Things must be protected from malware and other security vulnerabilities.

Reliable testing of IoT devices is the only effective way to effectively address the above factors. The general approach to quality assurance when testing IoT devices is to reduce testing periods to launch a reliable product with faster market launch. Therefore, it is important to conduct testing from the beginning of the development phase to identify and correct deficiencies at an early stage.

Problem statement: It is necessary to test IoT devices in order to avoid improper operation, which can lead to fatal consequences (such as security problems, communication problems, failure of sensors or the device itself, etc.). To prevent such problems, it is necessary to thoroughly test the IoT devices and their network environment before use.

2. Review of existing solutions and their use

Embedded systems are used in a wide range of technologies in various industries. Some examples:

Cars. Modern machines usually consist of many computers (sometimes up to 100) or embedded systems designed to perform various tasks in the vehicle. Some of these systems perform basic utility functions, while others provide entertaining or user-oriented functions. Some embedded systems in consumer cars include cruise control, backup sensors, suspension control, navigation systems and airbag systems.

Mobile Phones. They consist of many embedded systems, including graphical user interface software and hardware, operating systems (OS), cameras, microphones, and USB I / O modules (universal serial bus).

Industrial machines. They may contain embedded systems, such as sensors, and may themselves be embedded systems. Industrial machines often have built-in automation systems that perform specific control and management functions.

Medical equipment. They may include embedded systems such as sensors and controls. Medical equipment, like industrial machines, must also be very user-friendly so that machine errors that can be prevented do not endanger human health. This means that they will often include more complex operating systems and a graphical interface designed for the corresponding interface.

There are many examples of embedded systems. So, the article will show several examples and analyze their use, as well as their structure.

Automotive Embedded Systems:

Electronic control units are used in automotive embedded systems. This unit contains a microcontroller, switches, sensors, drivers, etc. All sensors and actuators are connected to the electronic control unit. Cars that use embedded systems can consist of hundreds of microprocessors. Each microcontroller performs its own special task. Some of them control the engine. Some launch the dashboard device. The whole system actually consists of several small systems. The use of embedded systems in the automotive industry has reduced the cost factor. This has improved overall performance and increased functionality. It has also reduced weight and made cars safer and more reliable. Applications of automotive embedded systems include:

- Automatic stability control
- Traction control system
- Pre-emergency safety system
- Airbag
- Car navigation system

So you can see that embedded systems have improved cars and made them more comfortable. Also, they have increased the functionality of cars and made them easy to use.

Home Security System:

Home security systems are widely used today. These systems have several functions, such as checking for fire or gas leaks, as well as detecting attempts by a suspicious person to enter the house. The microcontroller is used to control all operations. Sensors give data and if something wrong happens then

safety alarms get activated. Sensors used in such systems include gas sensors, smoke sensors, temperature sensors, IR sensors, etc. Such systems also include a keyboard for entering passwords on the gate. If the correct password is entered, this built-in system opens the gate, and if someone tries to enter the wrong password, the alarm goes off and the gate remains closed. The output signal comes from alarms or any display. The conclusion can also be sent to a remote location. If family members are not at home, they can still monitor what is happening in their home. The home security system is not limited to homes. Such systems can be used in stores, shops and in production. Almost every industry and office has security systems that can recognize workers by their faces or IDs. The home automation system is also one example of embedded systems as a home security system.

3. What is an embedded system?

3.1 Embedded systems

An embedded system is a computer system with a specific function in a larger mechanical or electrical system. They control many commonly used devices. They consume little energy, are small in size and their cost is low per unit. Modern embedded systems are often based on microcontrollers. A microcontroller is a small computer on a single integrated circuit that contains a processor core, memory, and programmable I/O peripherals. Because the embedded system is designed to perform certain tasks, they can be optimized to reduce the size and cost of the product, as well as increase reliability and performance.

Embedded systems have revolutionized science. It is also part of the Internet of Things (IoT), a technology in which objects, animals, or people are given unique identifiers and the ability to transmit data over a network without requiring human-to-human or human-to-computer interaction.

3.2 How embedded systems work

Embedded systems always function as part of a single device - this is what is meant by the term "embedded". These are inexpensive, low-power, small computers built into other mechanical or electrical systems. They typically consist of a processor, power supply, memory, and communication ports. Embedded systems use communication ports to transfer data between the processor and peripherals - often other embedded systems - using a communication protocol. The processor interprets this data using minimal software stored in memory. Software is usually highly dependent on the function performed by the embedded system. The processor can be a microprocessor or a micro controller. Microcontrollers are just microprocessors with peripheral interfaces and built-in memory. Microprocessors use separate integrated circuits for memory and peripherals instead of including them in the chip. Both can be used, but microprocessors usually require more support circuits than microcontrollers because they are less integrated into the microprocessor.

3.3 The structure of embedded systems

Embedded systems vary in complexity, but usually consist of three main elements:

- ***Hardware***. The hardware of embedded systems is based on microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers and usually refer to a CPU (CPU) that is integrated with other basic computing components, such as memory chips and digital signal processors (DSP). These components are built into the microcontrollers on a single chip.
- ***Software and firmware***. Embedded software software can vary in complexity. However, industrial-grade microcontrollers and embedded IoT systems typically run very simple software that requires a small amount of memory.
- ***Real time operating system***. They do not always work in embedded systems, especially in smaller systems. RTOS determine how the system works by controlling the software and setting rules during program execution.

In terms of hardware, a basic embedded system would consist of the following elements:

- Sensors convert physical sense data into an electrical signal.
- Analog-to-digital (A-D) converters change an analog electrical signal into a digital one.
- Processors process digital signals and store them in memory.

- Digital-to-analog (D-A) converters change the digital data from the processor into analog data.
- Actuators compare actual output to memory-stored output and choose the correct one.

The sensor reads external inputs, the converters make that input readable to the processor, and the processor turns that information into useful output for the embedded system.

3.4 Types of embedded systems

There are several basic types of embedded systems that differ in their functional requirements. Among them are:

- ***Mobile embedded systems*** are small systems designed for easy use and everyday portability. An example of this is digital cameras.
- ***Network-embedded systems*** are connected to the network to provide output to other systems. Examples are home security and point of sale (POS) systems.
- ***Standalone embedded systems*** do not depend on the main system. Like any embedded system, they perform a specialized task. However, they do not necessarily belong to the host system, unlike other embedded systems. An example of this is a calculator or MP3 player.
- ***Embedded systems in real time*** give the desired result after a certain time interval. They are often used in the medical, industrial and military sectors because they are responsible for the most important tasks that depend on time. An example of this is the traffic control system.

Embedded systems can also be categorized by their performance requirements:

- ***Small-scale embedded systems*** often use no more than an 8-bit microcontroller.
- ***Medium-scale embedded systems*** use a larger microcontroller (16-32 bit) and often link microcontrollers together.
- ***Sophisticated-scale embedded systems*** often use several algorithms that result in software and hardware complexities and may require more complex software, a configurable processor and / or a programmable logic array.

There are several common architectures of embedded system programs that become necessary as embedded systems grow and become more complex in scale. They include:

- ***Simple control cycles*** caused by routines that control a specific piece of equipment or firmware.
- ***Interrupt control systems*** have two cycles: primary and secondary. Interrupts in cycles cause tasks.
- ***Cooperative multitasking*** is essentially a simple control loop located in the Application Programming Interface (API).
- ***Warning multitasking or multithreading*** is often used with real-time operating systems (RTOS) and has strategies for synchronizing and switching tasks.

To sum up: Today, any embedded system is an IoT device that connects to the Internet and can communicate with other IoT devices. Also, the proposed complex allows you to test the network environment of the embedded system.

4. Types of test documentation and methodology

4.1 Test documentation

The basis of testing is knowledge and ability to write test documentation, as well as knowledge of different testing methodologies. There are three main types of documentation: test case, defect report, and troubleshooting. There are also different approaches and testing methods. Examples based on the TCP protocol stack will be shown. Let's take a closer look at the test documentation, which is aimed at helping developers and testers improve the product being developed.

Test case: A test case is a set of actions performed on a system to determine whether it meets the requirements of the software and is functioning properly. The purpose of the test case is to determine whether the various functions in the system work as expected and to confirm that the system meets all relevant standards, guidelines and customer requirements. The process of writing a test case can also help detect errors or defects in the system. Test cases are typically written by members of the quality assurance team (QA) or testing team, and can be used as step-by-step instructions for each system test.

Defect report: This is a document that identifies and describes a defect detected by a tester or user. The purpose of the defect report is to state the problem as clearly as possible so that the developers can easily reproduce the defect and correct it. Writing software bug reports is an important skill for software testers, quality control. The defect report contains a sequence of actions that leads to an unexpected result, or the user receives an unexpected error: the system hangs, or the calculated data is inaccurate, and so on.

***Troubleshooting*:** Troubleshooting is the process of diagnosing the source of a problem. It is used to troubleshoot hardware, software, and many other products. The basic theory of troubleshooting is that you start with the most general (and often the most obvious) possible problems and then narrow them down to more specific issues.

Few testing methodologies:

Performance testing - testing, which is carried out in order to determine how quickly a computing system or part of it works under a certain load. It can also be used to check and validate other attributes of system quality such as scalability, reliability, and resource consumption. This method will be described in more detail in Section 6, using the TCP protocol stack as an example.

Security testing: Application security for IoT devices is probably the biggest problem for most users. People can use applications for IoT devices for a variety of reasons, including monitory transactions, sharing personal information, buying and selling, and more. That's why the Internet of Things app needs to have special access control tools for users that restrict access to outsiders. Without security testing, it is impossible to detect major vulnerabilities in the application of the Internet of Things device and make sure that it does not transmit confidential information to attackers.

Required hardware and software: To test the TCP protocol stack, you need to have two computers with Linux installed, as well as a number of programs: wireshark (analog of the tcpdump traffic interceptor with a convenient and clear interface, iperf - TCP / UDP traffic generator, which allows you to check system stability, dhcp – server, and also some built-in Linux utilities).

4.2 Example of test documentation

Shown as an example one of the most popular types of test documentation, namely a defect report. A defect report is needed so that developers can quickly fix errors. Defect reports should include detailed information about the platform, environment, or other technical information to create detailed description. Your defect report should be clear and easy to read. The purpose of the defect report is to resolve the issue as soon as possible so that customers can continue to use the software. Here are the main sub-items for the report:

- Summary
- Description
- Build / platform
- Playback steps
- Actual results
- Expected results

This way, the developer or product manager can quickly get a clear idea of the error. The developer will be able to quickly reproduce the error using step-by-step instructions, and then correct it.

Here is an example of a report defect that describes a failed dhcp server startup and suggests a solution to this problem.

Summary: DHCP doesn’t start
OS: Ubuntu 20.04 Severity: Minor
Priority: Medium
Status: Assigned

Description: DHCP doesn’t start because dhcpd.conf isn’t configured correctly
Steps to Reproduce:

1. Configure dhcpd.conf
2. Start dhcp
sudo dhcpd
3. Check DHCP status
sudo systemctl status isc-dhcp-server

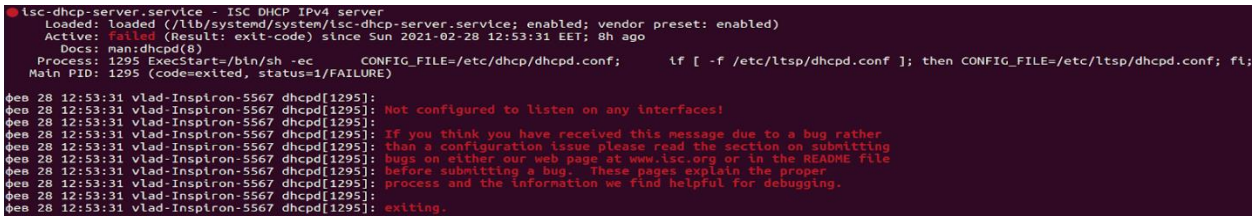

Fig. 1. Example of a failed DHCP server startup

Actual result: DHCP server failed to start due to incorrect dhcpd.conf configuration
Expected result: DHCP server started successfully

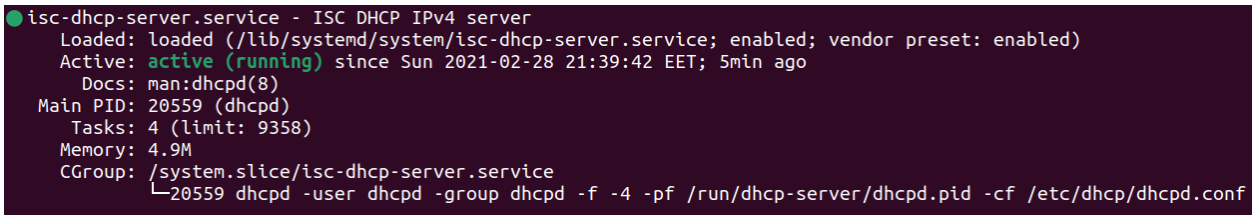

Fig. 2. An example of a successful start of a DHCP server

HowTo Fix: Configure dhcpd.conf right. In this case, write the subnet mask correctly

5. Results. Performance test

Performance testing is the practice of assessing how a system works in terms of response speed and stability under a given workload. Performance tests are usually performed to check the speed, stability, reliability and size of the program. The process includes such indicators of "effectiveness" as:

- Browser, page and network response time
- Server request processing time
- Simultaneous custom volumes are allowed
- CPU memory consumption; the number and type of errors that may occur in the application

In our case, testing the stability of the system will be demonstrated using bash scripts that are very easy to write and use. An example of such a scenario is shown below.:

```
#!/bin/bash
print_usage() {
echo "Usage: $0 ip_addr device"
        }
if [ $# -ne 2 ]; then
echo "Error: Too few arguments" >&2
print_usage
exit 1
fi
iperf -c $1 -i 1 -t 30 | tee tcp_$2.txt
```

This script uses an *iperf* traffic generator that requires a client and a server. This script accepts the IP address of the server and the client. You can also select the desired script execution time and the interval at which the result will be displayed. At the output we get a text file with a traffic log.

Next, you can use the *gnuplot* script, which can be used to draw a graph that will use *.txt traffic derived from the *bash* script. Further is an example of a *gnuplot* script:

```
set xlabel "Time, sec"
set ylabel "Mbit/sec"
```

```
files = system("ls -1 *.txt")
phone(f) = substr(f, 5, strlen(f) - 4)
plot                    \
for [file in files] file    \
using ($3>8 ? $7 : $8)   \
every ::6               \
title phone(file)       \
with lines
pause mouse close
```

You must make both scripts executable. To do this, use the command:
sudo chmod + x <file name>.
You can now run these scripts sequentially and get the result as a graph (fig. 1).

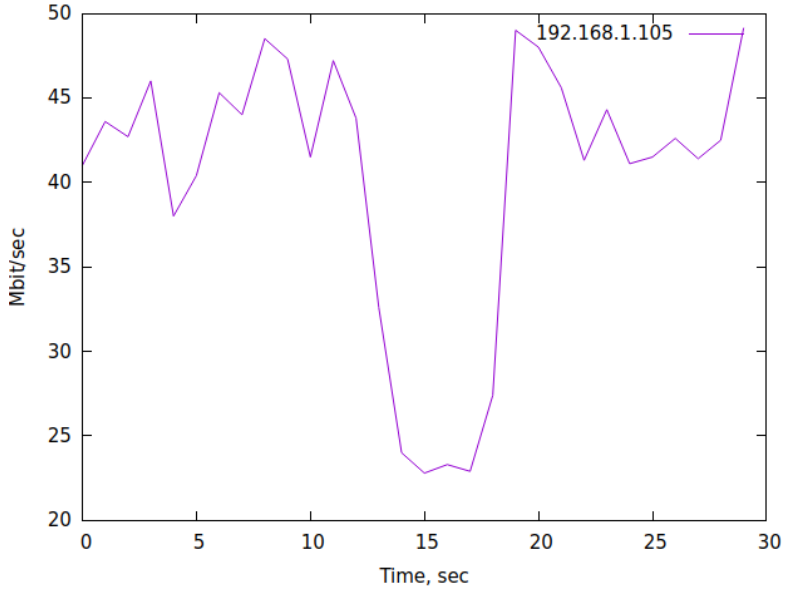

Fig. 3. Generated traffic graph

6. Conclusion

Today, the Internet of Things is one of the most popular industries. Therefore, in the development of IoT devices there is a need for detailed testing of these devices. If you do not fully test the device before selling it, many malfunctions may occur in the future, and this will result in the loss of a large amount of money for the company. Also this article discusses the basics of testing: writing test documentation, and some testing methods. The writing of the defect report on an example of start of the dhcp server was considered in more detail, and also the performance test which uses the usual bash script as a basis was demonstrated.

References

1. “Real Life Examples of Embedded Systems”. [Online]. Available: https://www.theengineeringprojects.com/2016/11/examples-of-embedded-systems.html
2. “What is an Embedded System?”. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/embedded-system
3. “How Embedded Systems Impact Your Everyday Life”. [Online]. Available: https://electronicsmaker.com/how-embedded-systems-impact-your-everyday-life
4. B. Ozer, “Smart cameras as embedded systems”. [Online]. Available: https://www.researchgate.net/publication/2955745_Smart_cameras_as_embedded_systems

5. “Embedded Systems - The Heart of Automotive Market”. [Online]. Available: https://www.techsciresearch.com/blog/embedded-systems-the-heart-of-automotive-market/44.html
6. “Importance of IoT Software and App Testing”. [Online]. Available: https://performancelabus.com/iot-testing-importance/
7. “The Internet of Things at home: Why we should pay attention”. [Online]. Available: https://www.computerworld.com/article/2490360/the-internet-of-things-at-home--why-we-should-pay-attention.html
8. “What is a Test Case?”. [Online]. Available: https://www.guru99.com/test-case.html
9. “What is a Defect Report”. [Online]. Available: https://techbeacon.com/app-dev-testing/write-software-defect-reports-get-results-boost-credibility
10. “Troubleshooting”. [Online]. Available: https://edu.gcfglobal.org/en/computerbasics/basic-troubleshooting-techniques/1/