Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 1 (2024)

Implementation of Fault-Tolerance of Computer Networks

Kornienko D. V. Mishina S. V.
Bunin Yelets State University

Abstract: The presence and global expansion of computer networks over the past decades has become commonplace. From building small local networks within the city district, they very quickly moved on to organizing a global network. Naturally, with the expansion of the zone of influence, the problems of both small and large communication providers have greatly increased. Most of them are solved exclusively by strictly following the instructions and unquestioningly observing the limitations and features of data transfer protocols. However, networks are built by people who are extremely prone to making mistakes for various reasons beyond their control. These errors lead to significant problems both in terms of customer-oriented and economic policies of companies. In this article, we have considered one of the options for solving the problems associated with the human factor in the maintenance and modernization of computer networks through the use of software based on the high-level Python programming language and the specialized Kivy framework. The main task set during the development of the terms of reference was the creation of cross-platform specialized software for the employees of the company directly involved in the construction and use of the network, capable of completely or significantly leveling the problems of the human factor while at the facility and reducing the time of network outages at the time of technical work. Despite the simplicity in design and implementation, data were obtained that characterize a significant reduction in the time of work, as well as a significant reduction in the number of errors when rebuilding network segments.

Keywords*: computer networks, human factor problems, network fault tolerance, automation and notification tools, Python, Kivy.*

1. Introduction

For these purposes, a diverse arsenal of hardware and software is used. Some of them support the stability of operation at the hardware level in order to provide redundant redundancy of information transmission media or hardware computing facilities. These include redundancy of communication channels by organizing additional information-physical connections by laying a parallel cable. In terms of organizing the smooth operation of hardware, clustering of server hardware is also used, designed to create a single fault-tolerant network that distributes the load and is able to ensure full operability in case of significant problems, up to the complete shutdown of one or more (depending on their number) computers.

Speaking about software implementations of maintaining network health, we should mention the reserves of hot and cold replacement of failed server software components [6]. This solution, in turn, allows you to automatically or manually transfer information for processing to another physical server as quickly as possible in the event of a shutdown of the process that processes incoming information, or a complete stop of the necessary service. The second example of this type of implementation of increased fault tolerance is the constant monitoring of the health of critical processes of computing systems on specific computers, as suggested by the authors of research papers [7-10]. Thus, a separate service is launched on each cluster machine, which monitors all changes in the behavior of the child service in real time. In the event of incorrect operation, an unexpected shutdown, or an excessively long response time, the parent process restarts the child process until its operation is established.

Based on this, we get an almost complete set of software and hardware tools for implementing the fault tolerance of computing equipment and introducing a replacement in case it is impossible to maintain the stable operation of the existing network.

Based on the above, we have the following algorithm for the interaction of hardware-software complexes for monitoring and redundant systems:

– the stage of physical redundancy monitors in real time damage to information transmission networks. During quiet times, it can be used as a means of increasing the bandwidth to reduce packet transmission delays and increase public throughput. If any problems are detected in automatic or manual mode, all passing traffic is concentrated on a single physical channel. Since this entire process takes no more than a second, the subscriber experiences inconvenience only in a short-term increase in response delays to ICMP requests [11];

– stage of software control and resumption of services. This stage serves exclusively for continuous monitoring of the correctness of the execution of tasks by running service services. In cases where a cable or other data transmission system is available and does not cause any failures, the overhead services required to provide various kinds of services may cause software failures due to various unforeseen circumstances. By monitoring their behavior, the specialized process has all the permissions to stop, start, and change the parameters of other processes. When a failed process is detected, an attempt is made to restart it. In the event of a denial of service, an unavailable process will be restarted with various startup and configuration options to resume full or high health. These attempts continue for a limited time, after which the signal is transmitted to the next stage;

– replacement stage. It implies the commissioning of another instance of a failed service on another physical platform or the full launch of a cold reserve, which is an identical configured copy of a computer with a parallel decommissioning of a problematic network node.

2. Problem

However, despite the abundance of automation tools, there is a predominant number of tasks that require direct monitoring and intervention by company employees. Such tasks, as a rule, imply direct intervention in the logical system of the network and its significant change. One of these tasks is the replacement of access level equipment, and in particular L2 switches. In addition to the temporary complete cessation of network operability in this area, there is an extremely high probability of errors associated with the human factor. During the work, situations may arise when the cable system is incorrectly connected or not connected at all.

3. Solution

In order to prevent such cases, we have written software that allows you to check the correctness of cable network switching at the access level section. The main task of this software is to save data on the physical addresses of subscribers on each port of the L2 level switch with their subsequent comparison with the results obtained after direct intervention in the cable subsystem. Since the performance of the application must be ensured both on stationary machines and on portable devices, it was decided to develop this software on a universal platform. By and large, a number of other software tools based on the ARP protocol could be taken as a basis, which are discussed in the article by Yuan-Cheng Lai, Ahsan Ali, Md. Shohrab Hossain, Ying-Dar Lin [12], written by other people. However, in our particular case, it is necessary not only to remember the presence of physical addresses, but also to provide functionality for storing, comparing and, most importantly, matching addresses to a specific switch port at a specified point in time, indicating errors. A large number of such software was viewed, but in the end, it was decided to write the program on our own.

Based on the research of Bilal Saeed, Tarek Sheltami, Elhadi Shakshuki [13], we chose the Python 3.10 programming language and the Kivy library for compiling GUI applications for Android and IOS for our project. This choice is due to the fact that Python is an extremely popular and easy-to-write scripting programming language. It is extremely versatile and for this reason is suitable for solving a wide variety of tasks on many software platforms: from IOS and Android to desktop and server operating systems.

In turn, the Kivy library is a free and open-source framework for developing mobile applications and other multi-touch application software with a natural NUI user interface. It is distributed under the terms of the MIT license and can run on all known operating systems.

As a first step, we started developing console software for desktop computers, since within the company's workstations there will be no need to install additional modules or compile the application. The general algorithm of the program is as follows:

– connection check block. At this stage, various checks are made for the availability of both public networks and service switch control networks based on examples from the work of Cüneyt Bayılmış, M. Ali Ebleme, Ünal Çavuşoğlu, Kerem Küçük, Abdullah Sevin [14];

– connection block. The next check of the availability of a particular switch in case of unavailability - the output of tips for connecting and direct entry into the console interface of the switch;

– maintaining the original state of the ports. Record mac-addresses, vlan and port number;

– saving the state of the ports after making changes. A step similar to the previous one;

– a block for comparing results and displaying an error message. The final step is to verify the records before intervention in the cable system and after. Next, you need to display an error report indicating the specific place in which you need to make changes.

4. Solution. Connection check.

The implementation of the first point is mostly based on ICMP requests to various network resources. Thus, pinging the DNS servers of various global companies allows you to verify that you have access to the global network (Listing 1). Similar requests to specific domains on the network indicate the health of their own DNS servers. The final step will be the direct organization of ICMP requests to the corporate network, which will allow you to check access with the ability to control.

```
def loading(self):
    Thread(target=self.running, daemon=True).start()
def running(self):
    self.get.disabled = True
    self.monitor.font_size = 30
    self.monitor.text = 'Запуск потока...[color=6df500]OK[/color]'
    response = os.system("ping -c 5 " + '8.8.8.8')
    if response == 0:
            self.monitor.text = self.monitor.text + '\nВыход в интернет...[color=6df500]OK[/color]'
```

Listing 1 - Checking WAN connectivity

It should be noted that when organizing such events, it is impossible to use single values for each stage of checks. In other words, each request must be duplicated to a similar service in order to validate even if the requested node itself fails.

5. Solution. Setting up connections.

The initial part of the second paragraph is largely identical to the first, except that ICMP requests go directly to the switch in order to check its availability (Listing 2). If a positive result is obtained as a result of the check, a function comes into play, the only task of which is to connect to the L2 switch using a telnet session. To

implement this functionality, the telnetlib library was used. After a successful connection is established, the user is given a choice of two options (out of three in the mobile version of the application).

```
host = str(self.host.text)
response = os.system("ping -c 5 " + host)
if response == 0:
    self.monitor.text = self.monitor.text + '\nПодключение к коммутатору...[color=6df500]OK[/color]'
    connect = telnetlib.Telnet(host)
    connect.write(b'login\n')
    time.sleep(1)
    connect.write(b'pass\n')
    time.sleep(1)
    count = 1
    os.chdir("/storage/emulated/0")
    if not os.path.isdir("MRDtemp"):
        os.mkdir("MRDtemp")
        self.monitor.text = self.monitor.text + '\nНаличие временных файлов...[color=f9301e]False[/color]'
    self.monitor.text = self.monitor.text + '\nПерезапись временных файлов...[color=6df500]OK[/color]'
    file_after = open('/storage/emulated/0/MRDtemp/' + host + '.txt', 'w')
```

Listing 2 - Checking corporate network connectivity and creating temporary files

6. Solution. Saving state.

The first option is to keep the state of the switch in its original state. In other words, recording physical addresses before they can be changed (Listing 3). The implementation of this action is organized as follows:

– first, the script analyzes what type of switch it is connected to. This is necessary to enter the exact value of the number of ports into the loop variable;

– when the quantity is determined, a loop is launched that buffers all the information from the switching table;

– then all received information is processed using regular expressions. This solution contributes to bringing the result into a convenient standardized form. In addition, the problem of subsequent comparison of different data is solved, since different switches have a different number of spaces and other special characters. When the received information is converted to the appropriate form, empty lines are removed, the entire record is saved to a temporary local file in the smartphone's memory.

```
self.monitor.text = self.monitor.text + '\n Starting a scan cycle...[color=6df500]OK[/color]'
while count < 25:
    zapros = 'show fdb port ' + str(count) + '\n'
    zapros_bytes = bytes(zapros, 'utf-8')
    connect.write(zapros_bytes)
    time.sleep(1)
```

```
            count = count + 1
            self.progress.value = count * 4
        mac = (connect.read_very_eager())
        mac_str = mac.decode('UTF-8')
        file_after.write(mac_str)
        file_after.close()
        file1 = open('/storage/emulated/0/MRDtemp/' + host + '.txt', 'r')
        file2 = open('/storage/emulated/0/MRDtemp/' + host + '_mac.txt', 'w')
        while True:
          line = file1.readline()
           if not line:
              break
           result = re.findall(r'\w+\s+\w\w-\w\w-\w\w-\w\w-\w\w-\w\w-\w\w\s+\d+', line)
           if len(result):
              result = str(result) + '\n'
              result = result.replace("['", "")
              result = result.replace("']", "")
              #result = result.replace(" ", " | ")
              file2.write(result)
        connect.write(b'logout\n')
        file1.close()
        file2.close()
        self.monitor.font_size = 30
        self.monitor.text = '[color=6df500]Запись состояния коммутатора произведена!'
        self.proverit.background_color = [1, 97, 1, 1]
        self.progress.value = 0
    else:
        self.monitor.font_size = 30
        self.monitor.text = 'Коммутатор [color=f9301e]НЕ[/color] доступен!\n\n(отсутствует подключение к оффисной сети через VPN\n или коммутатор не отвечает)'
        self.get.disabled = False
```

Listing 3 - Recording MAC Addresses from Switch Ports

7. Solution. Comparison of results.

The second possible point, in turn, ensures the receipt of data after the completion of the assigned work. The principle of its operation is absolutely similar to the above, with the exception of saving information to a second temporary file, which will protect information from possible destruction and provide an opportunity to

transfer it to the system administrator if necessary. When the information is saved, the comparison function is started. Its immediate task is to simply compare the two received files and then output the lines in which the data has been changed. Because this software implementation is intended for advanced system administrators, no further action is taken.

In the case of using the mobile version, in addition to the graphical implementation of the program interface, which is based on the syntax of the Kivy module, a third main menu item has been added, which allows not only to display the difference between compared files, but also to analyze changes and display problematic ports in a convenient graphical form (Listing 4). Thus, when you press the corresponding key, a separate program window opens, which schematically displays all the ports of the switch. In the normal state, each of them is marked in gray. Depending on changes in local files, the program classifies problems into two categories, each of which is assigned a corresponding color. When problems are detected, the port indicator on the smartphone is also highlighted in the corresponding color, and below is displayed specific information about the error and advice on how to fix it. For convenience, a re-check soft button has also been added to the graphical representation window.

```
class Second Window(Screen):
    def loading3(self):
        Thread(target=self.proverka, daemon=True).start()
    def monitor(self):
        self.graf_monitor.text = ''
        self.p1.background_color = [1, 1, 1, 1]
        self.p2.background_color = [1, 1, 1, 1]
        self.p3.background_color = [1, 1, 1, 1]
        ...
        file1 = open('/storage/emulated/0/MRDtemp/different.txt', 'r')
        file2 = open('/storage/emulated/0/MRDtemp/different2.txt', 'w')
        while True:
            # считываем строку
            line = file1.readline()
            # прерываем цикл, если строка пустая
            if not line:
                break
            # выводим строку
            result = re.findall(r'\S\s\w+\s+\w\w-\w\w-\w\w-\w\w-\w\w-\w\w\s+\d+', line)
            if len(result):
                result = str(result) + '\n'
                result = result.replace("['", "")
                result = result.replace("']", "")
                file2.write(result)
        file1.close()
```

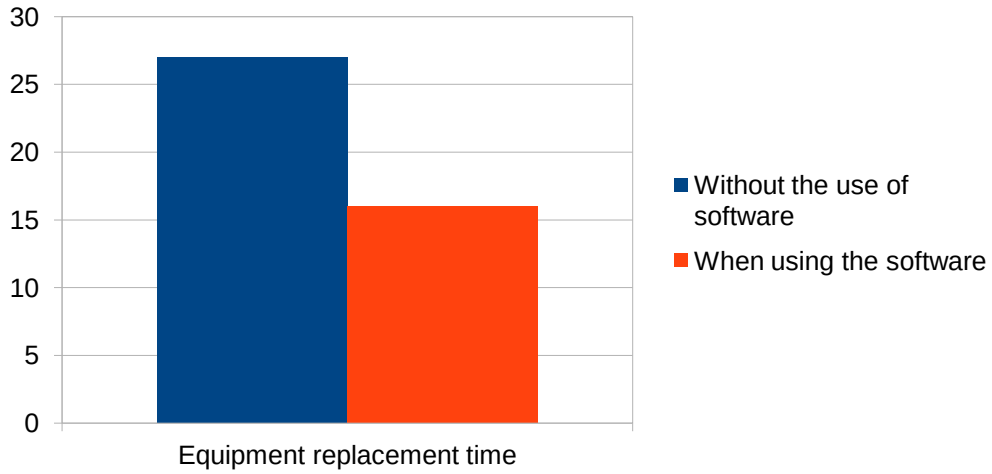
```
file2.close()
with open('/storage/emulated/0/MRDtemp/different2.txt') as f:
    text = '\n'.join(' '.join(line.split()) for line in f)
    file2 = open('/storage/emulated/0/MRDtemp/different.txt', 'w')
    file2.write(text)
   file2.close()
file4 = open('/storage/emulated/0/MRDtemp/different2.txt', 'w')
for line in open('/storage/emulated/0/MRDtemp/different.txt'):
    result = line.replace("- ", "-")
    file4.write(result)
file4.close()
file5 = open('/storage/emulated/0/MRDtemp/different.txt', 'w')
for line in open('/storage/emulated/0/MRDtemp/different2.txt'):
    result = line.replace("+ ", "+")
    file5.write(result)
file5.close()
with open('/storage/emulated/0/MRDtemp/different.txt') as file:
    for line in file.readlines():
        vlan = line.strip().split(' ')[0]
        mac = line.strip().split(' ')[1]
        port = line.strip().split(' ')[2]
        if '-' in vlan:
            self.graf_monitor.text = self.graf_monitor.text + '\n' + ('mac [color=f9301e]ПРОПАЛ[/color]! ' +
mac + " на порту: " + port)
            if port == '1':
                self.p1.background_color = [99, 1, 1, 1]
            if port == '2':
                self.p2.background_color = [99, 1, 1, 1]
            if port == '3':
                self.p3.background_color = [99, 1, 1, 1]
        …
```

Listing 4 - Generating a graphic display

At this stage, we can note the redundancy of temporary files and excessive detail. However, this is done intentionally for the purpose of debugging in case of file corruption when writing data from an inappropriate switch model.

8. Approbation and discussion

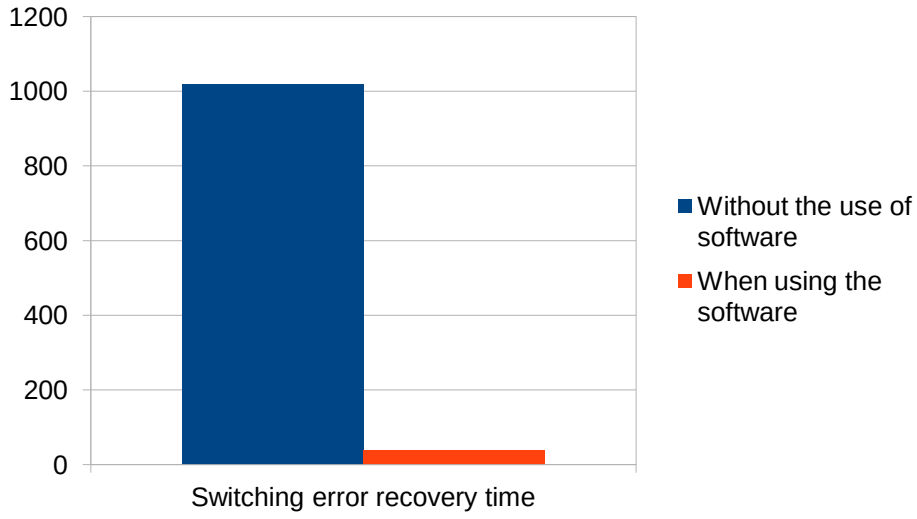
The final release of this software was tested in the organization of the regional Internet provider IT-Net LLC. As a result of the direct application of this software component, a number of advantages and disadvantages were identified and statistically processed. Speaking about the advantages, it should be noted the speed of work performed by engineers directly on the objects, as well as the improvement of the relationship between different divisions of the technical department, which is extremely important, according to research conducted by Alex Kesselman and Stefano Leonardi [15]. Thus, the amount of time required to replace access layer switches has been reduced by 60%. Figure 1 shows a graph that clearly shows the change in the time of the visit of engineers to the facility. It is worth noting that to keep a log of employee achievements, technologies for developing a secure RESTful web service API were used [16-17].



Source: Calculated by the author using engineering visit statistics

Figure 1 - Timeline for equipment replacement

In addition, the number of switching errors when replacing equipment remained the same, but the time to eliminate them decreased significantly. Figure 2 shows a timeline for error recovery in switching at the access layer.



Source: Calculated by the author using engineering visit statistics

Figure 2 - Time graph for elimination of switching errors

Speaking about the interaction of the personnel of the technical department, it should be noted that due to the use of the application directly by engineers, the number of calls to system administrators, network and lead engineers is reduced, which in turn allows the above personnel to increase the time spent on other tasks.

Turning to the disadvantages, we note a purely specific point related to the types of encryptions. In real use, OnePlus phones running Android 13 have been found to be unable to enable a connection to switches outside of IPsec. However, this problem is solved by setting up the equipment on the provider's side and a private problem of a particular phone. Therefore, within the framework of the article and the chosen organization, it can be neglected.

9. Conclusion

Thus, we solved one of the main problems, namely the human factor. No matter how good all technical means are, they need to be maintained by a person who, in turn, is extremely prone to errors. With this software solution, we have eliminated the possibility of a physical error, which ensures fast and high-quality maintenance of network equipment when replacing switches. Of course, this solution is not ideal and has some technical drawbacks. In addition to the problems of direct switching, a huge number of other factors also affect the time-of-service provision. But this solution made it possible to reduce the number of calls to subscribers due to a significant reduction in the time for both replacing equipment and eliminating errors in the performance of work. Among other things, the number of consultation calls from engineers to system administrators has decreased, which allows office staff to quickly complete a range of their immediate tasks, which in turn also improves response time to customer requests and their loyalty to the company.

The authors would like to thank the management of Bunin Yelets State University for financial support of this study.

Refrences

[1] Jiqing Chen, Yuanwei Jing. Multiple bottleneck topology TCP/AQM switching network congestion control with input saturation and prescribed performance.ISA Transactions.2023. Volume 135. Pages 369-379. https://doi.org/10.1016/j.isatra.2022.09.036;

[2] Ian W.C. Lee, Abraham O. Fapojuwo. Analysis and modeling of a campus wireless network TCP/IP traffic.Computer Networks. 2009. Volume 53, Issue 15. Pages 2674-2687. https://doi.org/10.1016/j.comnet.2009.06.002;

[3] Thomas Girdler, Vassilios G. Vassilakis. Implementing an intrusion detection and prevention system using Software-Defined Networking: Defending against ARP spoofing attacks and Blacklisted MAC Addresses. Computers & Electrical Engineering. 2021. Volume 90.106990. https://doi.org/10.1016/j.compeleceng.2021.106990;

[4] Seungpyo Hong, Myeungjin Oh, Sangjun Lee.Design and implementation of an efficient defense mechanism against ARP spoofing attacks using AES and RSA.Mathematical and Computer Modelling.2013. Volume 58, Issues 1–2. Pages 254-260. https://doi.org/10.1016/j.mcm.2012.08.008;

[5] Seung Yeob Nam, Sirojiddin Djuraev, Minho Park.Collaborative approach to mitigating ARP poisoning-based Man-in-the-Middle attacks.Computer Networks. 2013.Volume 57, Issue 18. Pages 3866-3884. https://doi.org/10.1016/j.comnet.2013.09.011;

[6] Jo Ann Oravec. Kill switches, remote deletion, and intelligent agents: Framing everyday household cybersecurity in the internet of things. Technology in Society. 2017. Volume 51. Pages 189-198. https://doi.org/10.1016/j.techsoc.2017.09.004;

[7] Sayed Qaiser Ali Shah, Farrukh Zeeshan Khan, Muneer Ahmad. The impact and mitigation of ICMP based economic denial of sustainability attack in cloud computing environment using software defined network. Computer Networks. 2021. Volume 187.107825. https://doi.org/10.1016/j.comnet.2021.107825;

[8] Mihailo Vesović, Aleksandra Smiljanić, Dušan Kostić.Fast and scalable routing protocols for data center networks.Digital Communications and Networks. 2022. https://doi.org/10.1016/j.dcan.2022.06.010;

[9] Benamar Bouyeddou, Benamar Kadri, Fouzi Harrou, Ying Sun.DDOS-attacks detection using an efficient measurement-based statistical mechanism.Engineering Science and Technology, an International Journal. 2020. Volume 23, Issue 4. Pages 870-878. https://doi.org/10.1016/j.jestch.2020.05.002;

[10] David Schneider.The state of network security.Network Security. 2012. Volume 2012, Issue 2. Pages 14-20. https://doi.org/10.1016/S1353-4858(12)70016-8;

[11] Prabha Shastri, Babu R Dawadi, Shashidhar R Joshi. Intelligent approach to switch replacement planning for Internet service provider networks. Sustainable Futures. 2020. Volume2.100036. https://doi.org/10.1016/j.sftr.2020.100036;

[12] Yuan-Cheng Lai, Ahsan Ali, Md. Shohrab Hossain, Ying-Dar Lin. Performance modeling and analysis of TCP and UDP flows over software defined networks. Journal of Network and Computer Applications. 2019. Volume 130. Pages 76-88. https://doi.org/10.1016/j.jnca.2019.01.010;

[13] Bilal Saeed, Tarek Sheltami, Elhadi Shakshuki. A Network Topology Discovery Tool for Android Smart Phones. Procedia Computer Science. 2015. Volume 63. Pages 104-111. https://doi.org/10.1016/j.procs.2015.08.318;

[14] Cüneyt Bayılmış, M. Ali Ebleme, Ünal Çavuşoğlu, Kerem Küçük, Abdullah Sevin. A survey on communication protocols and performance evaluations for Internet of Things. Digital Communications and Networks. 2022. Volume 8, Issue 6. Pages 1094-1104. https://doi.org/10.1016/j.dcan.2022.03.013;

[15] Alex Kesselman, Stefano Leonardi. Game-theoretic analysis of Internet switching with selfish users. Theoretical Computer Science. 2012. Volume 452. Pages 107-116. https://doi.org/10.1016/j.tcs.2012.05.029;

[16] D V Kornienko, S V Mishina, S V Shcherbatykh and M O Melnikov. Principles of securing RESTful API web services developed with python frameworks. *Journal of Physics: Conference Series.* 2021, 2094(3), 032016. https://doi.org/10.1088/1742-6596/2094/3/032016

[17] D V Kornienko, S V Mishina and M O Melnikov. The Single Page Application architecture when developing secure Web services. Journal of Physics: Conference Series. 2021, 2091(1), 012065. https://doi.org/10.1088/1742-6596/2091/1/012065

[18] Raparthi, M., Dodda, S. B., & Maruthi, S. (2023). Predictive Maintenance in IoT Devices using Time Series Analysis and Deep Learning. Dandao Xuebao/Journal of Ballistics, 35(3). https://doi.org/10.52783/dxjb.v35.113