«Сейфуллин оқулары – 16: Жаңа формациядағы жастар ғылыми – Қазақстанның болашағы» атты халықаралық ғылыми-теориялық конференциясының материалдары = Материалы Международной научно-теоретической конференции «Сейфуллинские чтения – 16: Молодежная наука новой формации – будущее Казахстана». - 2020. - Т.I, Ч.3 - С.130-132

THE DEVELOPMENT OF THE TCP PROTOCOL IN PYTHON

Zholdangarova G.I.

The article discusses the main transport layer protocols: TCP and its advantages. We analyze the logic of protocols and situations in practice using the Python programming language, in which it is preferable to use one or another type of blocking structure by the server. The TCP/IP stack also matches the OSI model and examples in the Python programming language that use these protocols.

Data transfer brings people together, and thus provides a reliable connection between them. One user can use various applications and services, such as Internet resources, games, and messengers, which are used for data exchange. The data is Packed and sent to the user we need.

The information transfer process is performed using the OSI layer. The processes described in the OSI transport layer receive data from the application layer and prepare it for forwarding on the network layer. the sender Computer communicates with the recipient computer to determine how to divide data into segments, how to prevent data loss, and how to verify the delivery of all data.

Transmission using TCP is similar to sending packets whose path is tracked from the sender to the recipient, and they are reliable. TSR uses the following basic operations to ensure reliability.

Tracking the number of segments sent to a particular node by an application.Подтверждение полученных данных.

Retransmission of segments when data is lost after a certain waiting time.

To understand how the TCP Protocol works, you need to figure out how to use the reliability tools, as well as how they track communication sessions. The TCP Protocol also provides the following features, which will be tested in practice in the Python programming language. Opportunities:

Establishing a communication session in TCP is a connection-establishing Protocol. Before forwarding any traffic, the connection Protocol negotiates and configures a persistent connection (or session) between the source device and the destination device. A session allows devices to agree on the amount of traffic that can be forwarded at a given time, as well as carefully monitor data transfer between the two devices.

Reliability of delivery in network terminology, reliability means guaranteed delivery to the destination node of all data segments sent by the source node, without exception. For many reasons, one of the segments may be damaged or completely lost during transmission over the network.

Delivery in the same order since multiple routes with different data rates can be used in networks, their order may change during data delivery. Using the numbering and ordering of the segments, TCP can guarantee that they will be assembled in the correct order.

Flow control of data transmission resources of the network nodes, such as memory or processing power is limited. When the TCP Protocol receives information that these resources are being used too actively, it may require the sending application to slow down the data flow rate. To do this, the TSR regulates the amount of information transmitted by the source. The data flow control feature prevents data from being sent again if the receiving node's resources are overloaded.

In order to understand how the TCP Protocol works, we use the socket module in the Python programming language. Socket-an abstract object that represents the connection endpoint. Socket connects two endpoints, and serves as a bridge between the server and the client. In order to establish such a connection in Python, a program was written that initiates the case of a client-server connection. This program can be seen in figure 2 and figure 3.**import** socket

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind(('127.0.0.1', 8888))
sock.listen(5)

while True:
    try:
        client, addr = sock.accept()
    except KeyboardInterrupt:
        sock.close()
        break
    else:
        result = client.recv(1024)
        client.close()
        print('Message', result.decode('utf-8'))
```

Figure 2. The program for the server

Figure 2 shows a detailed description of the program. The program consists of strings and each string has its own function. In this program, we used the conditional while operator, which is infinite.

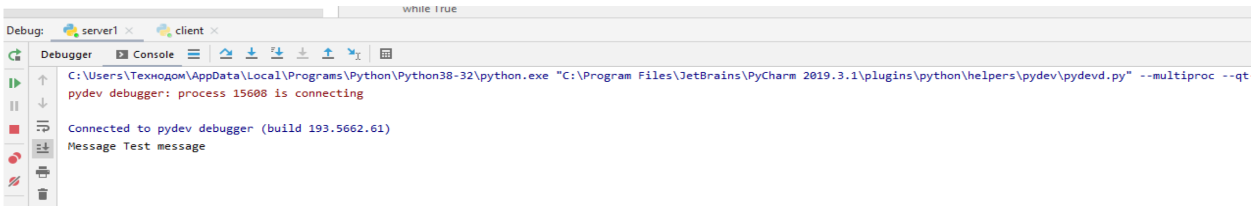
```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('127.0.0.1', 8888))
sock.send(b'Test message')
sock.close()
```

Figure 3. The program for the client

Figure 3 shows the client program, and the first, third, and fourth lines correspond to the description of the lead.

After running these programs, we get the result shown in figure 4. The shipment was successfully completed, so we were able to make sure that the TCP is really reliable.



```
C:\Users\Технодом\AppData\Local\Programs\Python\Python38-32\python.exe "C:\Program Files\JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\pydevd.py" --multiproc --qt
pydev debugger: process 15608 is connecting

Connected to pydev debugger (build 193.5662.61)
Message Test message
```

Figure 4. Result

Figure 4 in the console shows the result of the message we sent. The program passed the test, no losses were detected.

Based on the work done in Python, TSR met all expectations. Working with the help of a software language, we were able to thoroughly study the transport level of the OSI model. We could also see that the vehicle works without delays and losses. In a modern communication network, you can often encounter various problems, such as game freezes when using the UDP Protocol. I think that switching to TCP will solve many problems.

List of references

1. Gurikov, S.R. Fundamentals of algorithmization and programming in Python / S.R. Gurikov. - Moscow: Forum, 2018. - 991 c.
2. 6. Gurikov, S.R. Fundamentals of algorithmization and programming in Python. Textbook. Grif MO RF / S.R. Gurikov. - Moscow: Infra-M, Forum, 2018. - 707 c.
3. Zlatopolsky, D.M. Fundamentals of programming in Python / D.M. Zlatopolsky. Moscow: DMK Press, 2017. - 277 c.
4. Eric, Mathis Are Learning Python. Game programming, data visualization, web applications / Mathis Eric. - Moscow: Peter, 2018. - 760 c