UDC 519.688 **doi: 10.20998/2522-9052.2019.1.14**

H. Makogon[1], R. Musaiev[1], O. Dychko[1], I. Krylenko[1], Ye. Dumych[2]

[1] Military Institute of Tank Troops NTU "KhPI", Kharkiv, Ukraine
[2] Ukrainian Engineering Pedagogical Academy, Kharkiv, Ukraine

STUDYING OF THE INFORMATION EXCHANGE PROCESS IN OFFICE LOCAL NETWORKS BY USING MATHEMATICAL APPARATUS OF THE QUEUING THEORY

The **subject matter** of the article is the process of information exchange in the office local networks of institutions on computers with different technical parameters. The **goal** of the study is to analyze client-server interaction with socket development of an effective and secure communication program in the local network. The **tasks** to be solved are: to analyze the client-server interaction in the local network by using sockets, to create a code of communication in the local network for computers with different technical parameters using the mathematical apparatus of the **queuing theory,** to conduct research of the information exchange process in the system "client-server" in order to increase its effectiveness. General scientific and special **methods** of scientific knowledge are used. The following **results** are obtained. Today, the computer park used in the armed forces has a very large dispersion of technical parameters and installed software. Therefore, the organization of office local networks requires solving the scientific and technical task of implementing such an architecture that simultaneously meets the requirements of quality, speed, convenience and level of security of information exchange between computers with different Hardware and Software. It is expedient to use sockets for communication protocols allows simultaneous processing requests from a large number of users. At the same time, the administrator can adjust the settings at any time and redistribute ports on the servers. The use of the mathematical apparatus of the queuing theory enabled to conduct research of the information exchange process of the system "client-server" in order to increase its efficiency. Each server was described as a closed single-channel queuing system (QS) with unlimited idle time. **Conclusions**. The authors consider the implementation of the "client – server" architecture on the office local network, the successful solution of which will allow an organizations to work optimally and without interruption. The authors developed communication programme on the local network on computers with different technical parameters and tested the code to detect errors. The use of mathematical apparatus of the queuing theory enables to solve the tasks of structural and parametric synthesis of the local network of the institution and to work out the method of detecting distributed attacks on the network at the initial stage.

Keywords: client-server architecture; socket; local network; queuing theory; ordering; single-channel closed QS; informational security.

Introduction

Formulation of the problem and research tasks. Modern information technology has not only become widespread in the world, but it continues its "expansion" at a very fast pace. Their task is to process, monitor and record huge data streams with less time. Successful companies are trying to maximize automation of production, workflow, and more. It is not an exception to organizations and institutions.

Information infrastructure as one of the basic components of an institution's activity can be a guarantee of efficient document management, database work, state monitoring, etc. At the same time, its constant development not only facilitates the execution of current affairs, but also makes the structure more vulnerable in the field of information security. It is clear that the requirements of not only rational topology and resource intensity, but also cybersecurity are put forward to the local network in military appointment.

Today, the computer fleet used in most offices and offices has a very large dispersion of technical parameters. Accordingly, the software installed on computers is also different - from Windows XP to Linux Mint.

In addition, the structure of any database is created by certain parameters, and in such a way that the work with her in the regular mode is possible only with the use of special applications.

Consequently, the urgency of the work is conditioned by the fact that the organization of local networks in military institutions requires the solution of the scientific and technical task of implementing such an architecture that simultaneously meets the requirements of quality, speed, convenience and confidentiality of information exchange between computers with various Hardware and Software.

In this regard, the priority is to analyze the client-server interaction on the local network by using sockets, developing a program for communicating on a local network on computers with different technical parameters and the application of the mathematical apparatus of the theory of mass service for conducting research on the process of information exchange in the system "client – server" in order to increase its efficiency and security.

Methodological basis of the research were general scientific and special methods of scientific knowledge.

Analysis of recent research and publications. Todate Delphi client-server development literature related to the development and use of sockets is presented very little. Therefore, for a detailed study of this aspect, you can refer to the documentation for Linux and Unix-systems, because it describes in detail the technology deployment of applications using sockets, but, as a rule, in Perl or C++ [1, 2].

It should be noted that the analysis of the normative framework for the updating of the computer

© Makogon H., Musaiev R., Dychko O., Krylenko I., Dumich Ye., 2019

tanner park in institutions indicates a large inertia of this process [3–5].

Most sources provide a thorough statement of the theoretical basis for building local networks, but the issue of routing, remote work on computers, sharing of resources and information security of the mixed network of institutions is not sufficiently explored.

Typically, existing methods for building local networks do not take into account the individual values of various parameters characterize the work of the network and server, which significantly affect the effectiveness of detection of an attack and unauthorized access to server resources [6–18].

The **goal** of the study is to analyze client-server interaction with socket development of an effective and secure communication program in the local network.

Main material

1. Client-server interaction using socket

1.1 General information about sockets

The software for communicating on the local network with the information viewing on the server is proposed to develop the means of the environment of the Delphi software products and the programming language Object Pascal.

It is proposed structurally to organize a database, server and client parts.

Created software allows:

– to work on a local network basis;

– to control the work of information transmission ports;

– to review the information exchanged by users;

– to work on different operating systems.

The user's ability to connect to the server provided by the coincidence of the port used both by the server and the client. The purpose of this parameter is carried out in the property of Port. The connection type is determined by the ServerType parameter.

A server code is a component of a computing system that performs service (serving) functions on a client request, giving it access to certain resources or services. A simple, intuitive interface has been developed for the user to work with the software product.

OnClientWrite and OnClientRead events are used to send and receive information from the user. In this case, you can interact with the user through a parameter such as ClientSocket. When working it is proposed to use the following characteristics and property:

– the number of users connected at the moment;

– the number of active processes;

– the number of free processes;

– the port, hostname and local IP address;

– the unlocking and locking the socket.

Since the network can link different hardware platforms, it is proposed to agree on the formats of data to be transmitted, in particular, integer formats.

Two-byte integers are stored in memory in two consecutive bytes. In this case, there are two possible options: the first byte stores the lowest byte of the number, and in the second one, the older one, and vice versa. The storage method is determined by the hardware part of the platform. The network format of the representation of such numbers coincides with the format of the Motorola processor, that is, on platforms with the Intel processor need to rearrange the bytes when converting numbers into a network format.

Thus, created software can be used to securely transfer information on the local network of any establishment on computers with different technical parameters.

Obviously, the software itself will be more secure in terms of private communication and data privacy.

The authors tested the program for error detection. In the future, it is planned to extend the test-plan and write test-cases.

1.2 Client socket

Activating the client socket 1 (ClientSocket1) occurs when the user clicks the "Client" button on the "Send list request" button. The ClientSocket1 component connects to the server whose address is specified in the Address properties, and through the port specified in the Port property. After successfully establishing a connection with the server, you can start the data exchange between the client and the server. The client application will invoke the OnConnect event. The processor of this event is the procedure: "procedure TMainForm. ClientSocket1Connect (Sender: TObject; Socket: TCustomWinSocket); ". The body of the procedure consists of the line "Socket.SendText ('s');". The variable 's' is a request to obtain from the server a list of available video files.

On the receiving side, upon receiving data from the client, the OnClientRead event of the server socket 1 is executed, the procedure of which is the procedure: "procedure TMainForm.ServerSocket1ClientRead (Sender: TObject; Socket: TCustomWinSocket);". The body of the procedure begins with an analysis of the team came from the client. As the parameters of the constructor, the file name and the mode in which the file will be connected is transmitted. A text file named "output.txt" contains a list of video files available to the client. The fmOpenRead flag specifies the file read mode. For the correct reception of the file on the client side, the file size is firstly sent using the SendText method: "Socket.SendText ('Size:' + IntToStr (fs.Size) + # 0".

1.3 Server socket

It is reasonable to assume that the first step after creating a socket is to bind the socket of this protocol to its standard address with the bind function (SOCKET s, const struct sockaddr FAR * name, int namelen) ;. The second is the translation of the socket into listening mode by listen. Finally, the server must accept the client connection with accept or WSAAccept.

The s parameter specifies a local socket descriptor, created by the socket () function, on which client connections are expected.

The second parameter is a pointer to the structure in which the socket address is stored, which corresponds to the standards of the protocol used. It should be called struct sockaddr. The Winsock title file specifies the SOCKADDR type, which is the struct sockaddr structure. The third parameter specifies the size of the transferred address structure, which depends on the

protocol. If successful, bind () returns 0. In the case of an error, the value SOCKET_ERROR (ie -1).

To translate the socket into the state of waiting for incoming connections, the API function listen (SOCKET s, int backlog) is used;

The first parameter s is the socket descriptor.

The backlog parameter is the maximum number of incoming connection requests that can be queued for processing while the server is busy. The value of backlog depends on the protocol provider. Invalid value is replaced by the nearest allowed.

Call SOCKET accept (SOCKET s, struct sockaddr FAR * addr, int FAR * addrlen); Serves the first one in the queue connection request. Upon completion, the addr structure will contain information about the IP address of the requesting client, and the addrlen parameter is the size of the structure.

Parameter s is the connected socket in the listening state.

The second parameter is the address of the actual structure SOCKADDR_IN.

The addrlen parameter is a reference to the length of the structure SOCKADDR_IN.

In addition, accept returns a new socket descriptor corresponding to the accepted client connection. For all subsequent operations with this customer, a new socket should be used. Outgoing listening socket is used to receive other client connections and continues to be in listening mode.

Winsock 2 offers the use of the WSAAccept feature, which can establish a connection depending on the result of the calculation of the condition**.**

1.4 Socket code

Windows Sockets can be with the structure of any length (it depends on the protocol). For the protocols of the TCP / IP stack, the structure of sockaddr_in is used, the size of which is also 16 bytes. Of these, only eight are used: two for encoding a family of protocols, four for an IP address, and two for a port. The remaining 8 bytes are not used and must contain zeros. The sin_zero field must contain an array of zeros. This is the same field that does not carry any semantic load and serves only to increase the size of the structure to the standard 16 bytes. The sin_family field should have PF_Inet value. The sin_port field records the port number to which the socket is tied. The port number should be written in a network format, that is, here it is necessary to use the function HtoNS, so that from the usual port number of us to get the number in the right format. The port number can be left to zero - then the system will select for a socket a free port number from 1024 to 5000.

The IP address for socket binding is set by the sin_addr field, which has the type TInAddr. This type is itself a variant entry that displays three methods of assigning an IP address: in the form of a 32-bit number, in the form of four 8-bit numbers. In the socket library, the InAddr_Any constant is provided, which allows you to not specify the explicit address in the program, but leave it at the discretion of the system. To do this, the sin_addr.S_addr field must be set to InAddr_Any. If the computer's IP address is not assigned, using this constant socket will be linked to the local address 127.0.0.1. If a computer has multiple IP addresses, then one of them will be selected, and the bind itself will be delayed until the connection is established (in the case of TCP) or before the first sending of the data through the socket (in the case of UDP). The choice of a specific address, however, depends on which address the remote party has.

It is fundamentally for the server, which port it will use - if the port is not defined in advance, the client will not know where to connect. Therefore, the port number is an important sign for the server. In systems with multiple network cards, the binding of a socket to an address in the event that its choice is entrusted to the system may be performed not while performing the Bind function, and later, when the system becomes clear why this socket is used: when the TCP client connects to the server, the system at the address of this server determines which card the card must go for exchange and chooses the appropriate address. The same thing happens with the UDP client: when it sends the first datagram, the system at the address of the recipient determines which map to attach the socket to. Therefore, in this case, the client may leave the choice of address at the discretion of the system.

The system binds the UDP server socket to the address; it waits for the packet to be received. At this time, the system does not have any information about which nodes will be exchanged through this socket, and can choose not the address that you want. Therefore, sockets of UDP servers operating on such systems should be explicitly tied to the desired address. Sockets of TCP servers in the standby mode and having InAddr_Any address, allow connection to them from any network interface that is in the system. A socket created by such a server when connecting a client will be automatically linked to the IP address of the network interface through which the client interacts with it. Thus, sockets created to interact with different clients may be tied to different addresses. After successfully completing Socket and Bind functions, the socket is created and ready to work. Further actions with him depend on which protocol he implements and for which role is assigned.

When the socket is no longer needed, one must release the resources associated with it. This is done in two steps: first the socket "turns off" and then closes.

To exclude a socket, the Shutdown function with the following prototype is used: function Shutdown (S: TSocket; How: Integer): Integer;

The S parameter defines the socket to be turned off, the How parameter can take values SD_Receive, SD_Send, or SD_Both. The function returns zero in case of successful execution and Socket_Error in the event of an error. Calling the function with the SD_Receive parameter prevents reading data from the input socket buffer. However, at the protocol level, the call to this function is ignored: UDP datagrams and TCP packets sent to this socket continue to fit into the buffer, although the program can no longer pick them up.

When using the SD_Send parameter, the function prohibits sending data through a socket. When using the TCP protocol, the remote socket receives the special

signal provided by this protocol, informing that more data will not be sent. If at the time of the Shutdown call in the output buffer, data is first sent, and then only the signal about the completion. The UDP protocol does not allow such signals, so when using this protocol, Shutdown simply prevents the socket library from using the specified socket to send data.

The SD_Both parameter allows simultaneously to deny both reception and data transfer through a socket. To release socket-related resources, the CloseSocket function is used. This function frees the allocated memory for buffers and the port. Its single parameter specifies the socket to be closed, and the return value is zero or Socket_Error. After calling this function, the corresponding socket descriptor no longer makes sense, and can no longer be used.

By default, the CloseSocket function immediately returns the call control of its program, and the socket closure process begins to run in the background. Closure implies not only the release of resources, but also the sending of data that remained in the socket's output buffer. If the socket is closed with one thread at a time when another thread tries to perform some operation with this socket, then this operation ends with an error.

The Shutdown feature is needed first of all to inform the communications partner in advance of the intention to complete the connection, and this only makes sense for the protocols that support the connection. When using UDP, the shutdown function is almost meaningless, you can immediately call the CloseSocket. When using TCP, the remote party receives a signal to shutdown the partner, but the standard socket library does not allow the program to detect its receipt. But this signal may be important for intra system functions that implement sockets. The Windows version of the socket library refers to the lack of this signal quite liberally, so the Shutdown call, when both the client and the server are running Windows, is not required. But the implementation of TCP in other systems is not always as lenient to such negligence. The result may be a long (up to two hours) "hanging" state of the socket in the system, when it is no longer possible to work with it, and the error information is not received by the program. Therefore, when using TCP it is better not to neglect Shutdown, so that the socket on the other side has no problems.

Recommended offensive order zakrittya TCP socket. In the first place, the server is not guilty of storing your own socket for a powerful innotation, which can be done only if you have been using it for a private socket. The key is to close the socket to the Shutdown folder ... with the SD_Send parameter. The server will send a message to all the users, it's overloaded into the buffer of the socket of the client, and then it will send a message about the completion of the client's transmission. Tim is not lesser, the socket of the client's code is delivered to the host, to which the server, if necessary, can be sent to the client, whether it be sent to him or her, be it necessary. Witm Shutdown server with the SD_Send parameter, and CloseSocket. Clients prodovuzhit read dani z vididnogo socket buffer to silently, leaving the signal about the completion of the transmission server. Signing the key of such a wikio CloseSocket. Such a guarantee is a guarantor, but this will not be trapped.

2. Creating of interaction program in the local network

2.1 Tasks

It is proposed the interaction programme in the office local network with a functional satisfies the following requirements:

- working on a local network basis;
- controlling transmission of information between users;
- working on different operating systems.

2.2 Functional program

By the word "chat" we will mean an online resource with chat features, chat program.

The chat page is automatically updated with the specified periodicity.

Instead of periodically reloading the page, a socket is opened between the client and the server, allowing you to instantly send or receive messages, consuming less traffic. In general, the project program is used to communicate on a local area network.

2.3 Program interface

To work with a software product, the client is encouraged to use a simple, intuitive interface. When start the program, the main window appears (Fig. 2).
The toolbar has windows:

- to fill the address of the desired server;
- to enter the port address;
- to enter a personal name.

There is a panel for choosing message colors and buttons for connecting and disconnecting from the server.

In order to get started, you need to fill in the windows: server addresses, port addresses, personal names.

After that you can start working with the program.

After connecting to the server, the client can send messages (Fig. 1).

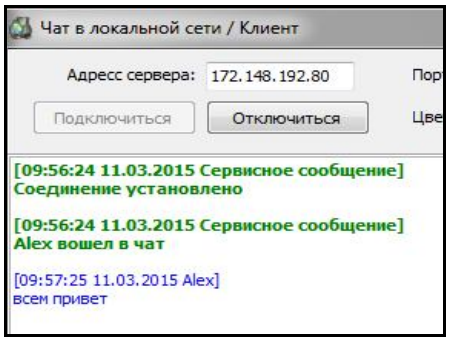
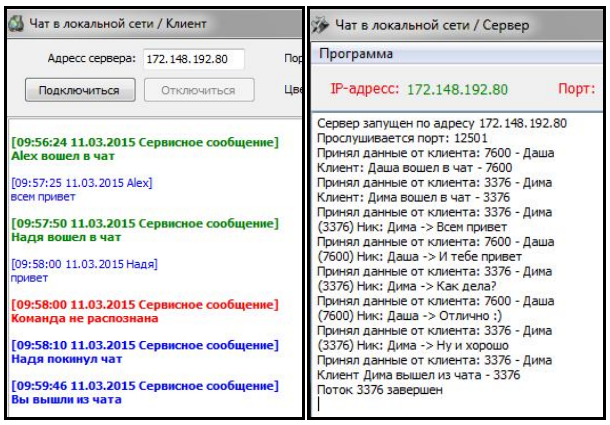


Fig. 1. Sending a message

After the correspondence client can easily exit the program (Fig. 2, a). When starting a program on behalf of the server, the user will see the following window (Fig. 2, b). The server user can view all correspondence between users of the local network (Fig. 2, c). Also in the server program there is a setup menu where you can set the startup time (Fig. 2, d) [19–25].



a – Exit the program c – Overlapping correspondence

b – Running the program d – Setup menu

Fig. 2. Windows of programm

3. Study of the process of information exchange in the system "client-server" to increase its efficiency

3.1 Defining a precedent diagram in the client-server system

A server program is a component of a computing system that performs service (serving) functions on a client request, giving it access to certain resources or services. We will conduct a study of the process of information exchange in the "client-server" system in order to increase its efficiency. For this we consider the diagram of precedents (Fig. 3).

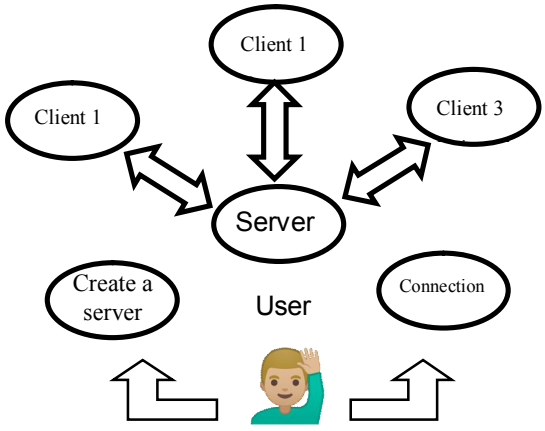


Fig. 3. Case diagram of the client-server system

The socket protocol allows to simultaneously process queries from a large number of users. At the same time, the administrator can adjust the settings at any time and redistribute ports on the servers. Thus, the number of simultaneously serviced users will be optimal, depending on the technical characteristics of the computer on which the server is located. From the point of view of the queuing theory each server is a closed single-channel queuing system (QS) with unlimited standby time (Fig. 4): *0* - requirements source, client; *1* - servicing device, server; *a* – input flow of service orders; *b* – outflow of processed orders.

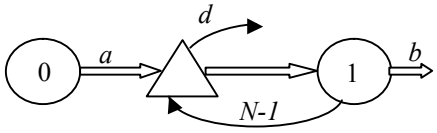


Fig. 4. The server as a closed single-channel queuing system (QS).

As can be seen from the figure, in this model there is a store of the input orders OA. If the order comes from the client 0 at the time when the OA is busy with the service of the pre-order, the new order gets into the order store on the input OA, but the time of placing the order in the drive is limited by some amount (timeout).

Let's determine: *(N-1)* - the number of orders that can fit in the drive; *d* - the order flow that throws the system out when waiting time is exceeded.

We will consider a plurality of TCP packets arrive at the server as an input flow of orders. Let's show that under certain conditions this flow can be considered Poisson. The intensity of this stream may depend on the time it takes to look at it for quite large intervals of time. For example, during the daytime during daytime, its intensity may be greater than at night. However, with a decrease in the duration of the analyzed interval, the intensity of incoming TCP orders is stabilized and can be considered as some constant value. For different networks, the length of such a gap may be different (typically from several minutes to several hours) and can be installed experimentally. In this case, the probability of receipt of k requirements in the interval $(0, t)$ is equal to the probability of receipt of k requirements in any other interval of the same duration $(a, a + t)$ within a given interval. Thus, the considered stream has the property of stationary.

Next let's assume users of the local network access to server resources independently of each other. If a single TCP connection is established on a single user request to the server, then the demand stream has the property of the absence of an aftereffect (according to the definition of this property).

3.2 TCP connection server as QS

The flow of TCP packets that arrive at the server, under given conditions, is Poisson. This means that it can be viewed as a coming requirements flow into the QS. An array of servicing devices will be considered server resources intended to store the parameters of TCP connections. In this interpretation of the service requirement, it is the reservation of the corresponding resources either to the successful establishment of the TCP connection, or until the end of the time-out server. Let's consider in more detail server resources serving as servicing devices. The parameters of the TCP connections are stored in the appropriate buffer represented as an array of dimension L, whose elements store the parameters of the TCP connections. They can be divided into three types: those containing the parameters of established connections, semi-open connections and free ones.

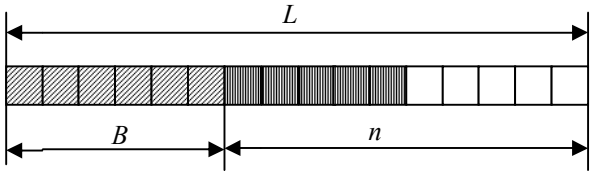
Let B be the number of TCP connections currently open. Then the number of the second and third type's

elements, the totality of which we will consider as an array of servicing devices QS, is determined by

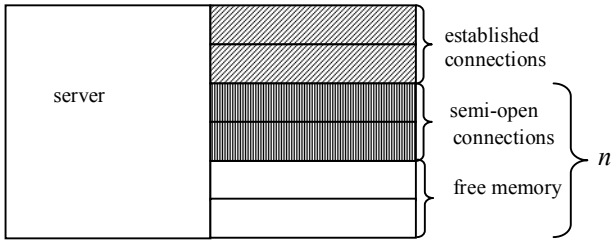
$$n = L - B. \quad (1)$$

In this case, the equipment occupied by service requirements is the elements of the second type.

In fig. 5, a it is depicted the described array, and in fig. 5, b server resources are provided in the winch of a set of servicing devices.



a – Buffer for storing TCP connection parameters



b – TCP server as an array of serviced devices

Fig. 5. Server resources sharing

Depending on the intensity ratio of the input flow of the requirements λ and the dimension of the array L, two types of QS can be considered. If the intensity of the input flow of orders is much less than the server's capabilities, which is the case for most modern systems, then it is expedient to consider the QS with an infinite number of servicing devices. Otherwise, one can consider the QS with failures. Given that, in practice, in the normal mode of operation of the server with a large margin cover the input requirements, the consideration of the system with failures is irrelevant. Then we will consider the system of the first type.

3.3 QS with an infinite number of servicing devices

As it has been shown, in order to describe the model of interaction between users and the TCP connection server in normal operation, it is advisable to consider QS with an infinite number of servicing devices. We denote the intensity ratio of the input flow of requirements λ to the average service time of the order μ by the coefficient $\alpha = \lambda / \mu$. Since the flow of requirements is Poisson, the probability that there are exactly k requirements in the system is defined as

$$p_k = \frac{\alpha^k e^{-\alpha}}{k!}. \quad (2)$$

The average number of devices employed (the total number of semi-open connections) is obtained as:

$$N = \sum_{k=1}^{\infty} k \cdot p_k = p_0 \sum_{k=1}^{\infty} \frac{\alpha^k e^{-\alpha}}{(k-1)!} = \alpha(1 - p_\infty). \quad (3)$$

In accordance,

$$p_\infty = \lim_{k\to\infty} \frac{\alpha^k e^{-\alpha}}{k!} = e^{-\alpha} \lim_{k\to\infty} \frac{\alpha^k}{k!} = 0. \quad (4)$$

From the relations (3) and (4) for QS with an infinite number of servicing devices we have:

$$N = \alpha(1 - p_\infty) = \alpha(1 - 0) = \alpha = 0 \quad (5)$$

The proposed model describes the server operation in the normal mode and allows considering such parameters as the requirements intensity to the server and the average time of the order service. However, such QS does not fully describe the server's operation, because it does not take into account the possibility of packets loss transmitted on a local network by users have different technical parameters. To improve the proposed model, it is expedient to divide the considered QS into two systems that serve orders for normal connection setup (when all packets are delivered) and semi-open connections that are removed by the timeout. To divide the output flow requirements into a plurality of orders, for each system, one must enter a criterion that allows you to determine the relevance of applications to the above types.

3.4 A model accounting the loss of packets in the local network

As noted above, the previously proposed model requires averaging average maintenance time for all requirements that does not fully take into account the features of the process of establishing TCP connections on the local network with users that have different technical parameters. To eliminate this disadvantage, we divide the above-mentioned HMI into two systems: QS_1 and QS_2. It'd be assumed that the first system describes the maintenance of applications for which half-open connections will be successfully installed after receiving the packet server, and the second, the requirements for which the connections will not be established and after the expiration of the waiting time will be deleted.

Let's accept that the time of exchange of pairs of packages between arbitrary sockets does not exceed the threshold. Provided that the client has correctly implemented the TCP protocol, the appearance of semi-open connections that are not installed within a given time interval (timeout) is due to the loss of the packet.

Therefore, to the requirements of the second type we will assign orders for which the TCP connection is in a semi-open state longer than. Let's denote by *s* and *l* the number of connections of the first and second types, respectively. This representation of the server is shown in Fig. 6.

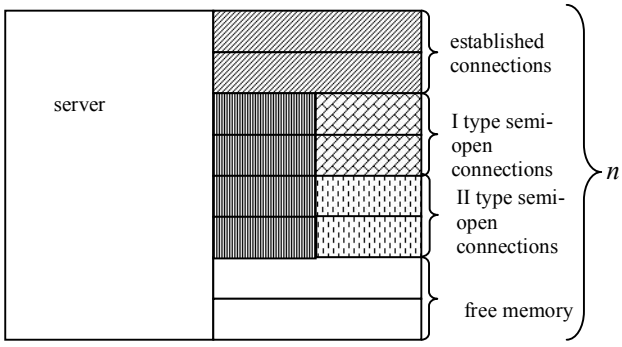


Fig. 6. TCP connection server as QS_1 and QS_2

Define relations that describe the state of such a system. Similarly (5) we define the average number of semi-open connections:

$$N = s + l = \alpha_1 + \alpha_2 = \lambda_1/\mu_1 + \lambda_2/\mu_2 . \quad (6)$$

Receipt intensity of orders-requirements is determined by the following ratio:

$$\lambda_2 = \lambda \cdot P_{no}, \quad (7)$$

where λ_2 □ the packet arrival rate of the network card of the TCP server; P_{no} □ the probability of a semi-discovered connection will not be installed.

The average number of such orders in service in the QS is determined by the second appendix of the formula (7):

$$l = \frac{\lambda_2}{\mu_2} = \frac{\lambda \cdot P_{nn}^{N_{SA}}}{\mu_2} = \frac{\lambda(2P_{nn} - P_{nn}^2)^N}{\mu_2}, \quad (8)$$

where μ_2 – timeout allocated on the server to establish a TCP connection; P_{nn} – the probability of packet loss in the network; N_{SA} – number of copies of SYN + ACK packets sent by the OS.

P_{nn} will be considered as a random variable characterizing the average number of occupied devices in the considered QS, and is conquered by the Poisson law of expansion. In our case, the parameter of this distribution is equal to 1. Mathematical expectation and dispersion are also taken equal to 1.

Let's believe that the parameter P_{no} depends on the quality of the network characterized by the probability of a packet loss on the network P_{nn} and is determined by the probability of an accidental event to spoil the packet on the network:

$$P_{no} = P_{nn} + (1 - P_{nn})P_{nn} = P_{nn} + P_{nn} - \\ -P_{nn}^2 = 2P_{nn} - P_{nn}^2. \quad (9)$$

In fig. 7 is the graphs of the distribution density and the distribution law of a random variable, conquered by a Poisson law with a parameter $\lambda > 10$.

3.5 Statistics of the SMO of the local network of users with different technical parameters

To effectively use the above methodology in practice, it must be possible to determine the actual values of the model output parameters for the system is in normal condition. As shown above, such parameters are: the intensity of the flow of applications, the likelihood of packet loss on the network to which the server is connected and the average service time of the application (successful installation of TCP connection).

The statistics of packet time, the most complete representation of the population, was obtained using the fsockopen procedure: "@ $ fp = fsockopen (" 192.168.1.1 ", 80, $ errno, $ errstr, 1); if (! $ fp) else mysql_close ();".

To accumulate ping-statistics, various remote computers with different operating systems were used. On this basis, it can be argued that the results obtained are quite generalized.

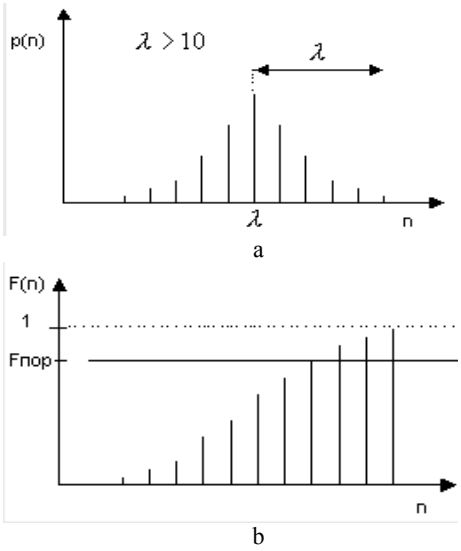


Fig. 7. Characteristics of the random variable, conquered by the Poisson distribution law with the parameter λ > 10: a – density of distribution; b – the law of distribution

The results obtained are illustrated below.

In Fig. 8 it is shown the orderly growth of return times statistics without taking into account the three packages, the return time of which is 40 times the return time of most packages.

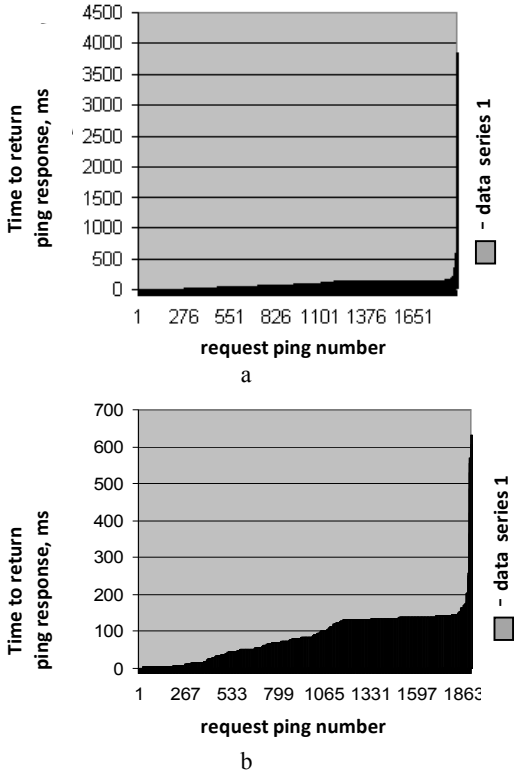


Fig. 8. Packet return time: a – normal scale; b – enlarged scale

In Fig. 9 one can see the empirical distribution of response time response times. On the abscissa the time intervals received by breaking the whole range of values in increments of 10 ms are postponed. On the axis of ordinates the number of values caught in the interval is postponed.

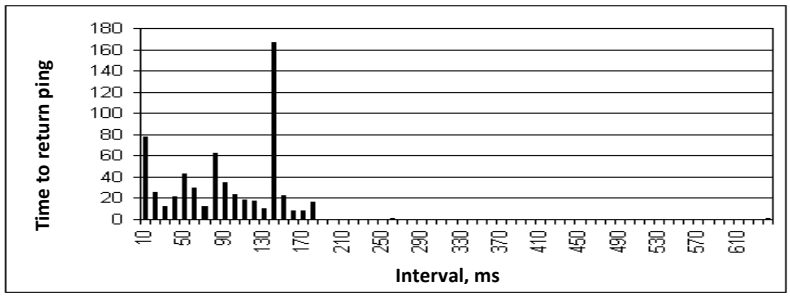


Fig. 9. Empirical distribution of return response time

According to the experimental data, the following value was obtained:

$$P_{nn} = \frac{4}{1904} = 0,0021 . \quad (10)$$

Under the Fig. 9 it is possible to put forward the hypothesis that the distribution of random variables is a composition of several distributions with different parameters, the definition of which is the prospect of development of the received methodology.

Note it is precisely the application of the theory of mass service that allows us to tune:

- task of parametric synthesis - definition of parameters of the system of service at its specified structure, depending on the parameters and properties of the messages flow, discipline and quality of service;

- the task of synthesizing the system structure with the optimization of its parameters in such a way given flows, discipline and quality of service, the cost of QS was minimal, or there were minimal loss of calls for given flows, discipline and cost of the system.

4. Applying queuing theory to reflect distributed attacks on the local network.

The specific share of the functional tasks of the local network installation is the organization of document circulation. Undoubtedly, the first use of such technologies is a significant increase in the efficiency of case management by accelerating the search for the necessary information, increasing the speed of information exchange, reducing the percentage of lost information, etc. Such means of cryptography as electronic digital signature can provide the provision of such basic information systems services as confidentiality, integrity of information, etc. [3]. Unfortunately, recently, a class of attacks aimed at denial of service has become more and more widespread. Successful implementation of such attacks allows blocking the access of users of information systems to the resources of different servers, which can remove the entire system from the working state**.**

On the basis of the proposed mathematical model that describes the maintenance of the server request flow for the establishment of TCP connection as a QS, the authors reviewed the method of detecting TCP attacks and finding the permissible values of the interval for the number of semi-open TCP connections on the server, working in the normal mode (for conditions of no attack). In accordance with this technique, the decision to start the attack is taken in the case when the real number of half-open on the server connections goes beyond the permissible interval.

To determine the limits of this interval, it is necessary to obtain values of such parameters as the intensity of the incoming application flow, the time during which the application will be served with a given probability, the probability of loss of IP packets for transmission over the network, the number of attempts to retransmit the packet server, the timeout allocated to the server for the installation TCP connection, and the likelihood of a true attack detection.

Part of the parameters can be determined using the proposed methods of data collection; part is determined by the settings of the TCP / IP protocol stack on the secure server.

Note that the collection of statistical data in addition to purely technical parameters were vrafovani LAN factors associated with human factors information security, socio-psychological specificity "client-server" communication and information culture level from [26–30].

It should be noted that in collecting statistical data, in addition to the purely technical parameters of the local network, factors related to the human factor of information security were considered: the socio-psychological specificity of client-server interaction and the level of information culture of users.

Conclusions

1. Today the computer fleet used in office local area networks has a very large dispersion of technical parameters and software installed on computers.

Therefore, the organization of office local networks requires solving the scientific and technical task of implementing such an architecture that simultaneously meets the requirements of quality, speed, convenience and level of security of information exchange between computers with different Hardware and Software.

2. The authors consider the implementation of the “client – server” architecture on the office local network, the successful solution of which will allow an organizations to work optimally and without interruption.

It is expedient to use sockets for communication protocols, which allows simultaneous processing requests from a large number of users.At the same time, the administrator can adjust the settings at any time and redistribute ports on the servers.

The authors developed communication programs on the local network on computers with different technical parameters and software, and tested the program to detect errors.

3. Using the mathematical apparatus of the **queuing theory,** to conduct research of the information exchange process in the system "client-server" in order to increase its effectiveness.

Each server was described as a closed single-channel queuing system (QS) with unlimited idle time.

4. Application of the queuing theory allows to solve the problems of structural and parametric synthesis of the local network of the institution. After simulating a local measurement operation organized for the method under consideration, it was discovered that the attack could be seen at an initial stage, and the socet connection is stable to a sharp increase of the input flow intensity of requests to server, taking into account characteristics both the network and the protected server.

References

1. Matthew, N. and Stones, R. (2009), *Beginning Linux Programming*, John Wiley [distributor], Indianapolis, Ind. Wrox, Chichester, the USA.
2. Lincoln, S. (2001), *Network Programming with Perl*, Williams, Moscow, 720 p.
3. Legislation of Ukraine (2013), № 635, *On Approval of Methodological Recommendations on Enterprise Accounting Policies and Changes to Certain Orders of the Ministry of Finance of Ukraine*, the Ministry of Finance of Ukraine, LIGA:ZAKON, Kyiv, avaiable at: http://search.ligazakon.ua/l_doc2.nsf/link1/MF13052.html.
4. Legislation of Ukraine (2014), № 25-14/677, Concerning the write-off of budgetary institutions of computer equipment, State Financial Inspection of Ukraine, LIGA:ZAKON, Kyiv, available at: http://search.ligazakon.ua/l_doc2.nsf/link1/FN004233.html.
5. Legislation of Ukraine (2007), № 2. Methodical recommendations on inventory accounting, approved by the Ministry of Finance Order, the Ministry of Finance of Ukraine, available at: http://search.ligazakon.ua/l_doc2.nsf/link1/MF07002.html.
6. Polyak-Braginsky, A. (2008), *Linux and Windows on the home network*, BHV-Petersburg, St. Petersburg.
7. Bogachev, K. (2003), *Basics of parallel programming*, BINOM, Moscow.
8. Date, K. (2001), *Introduction to database systems*, Williams, Moscow.
9. Kamer, D. and Stevens, D. (2002), "Development of client-server applications", *TCP/IP networks*, Williams, Moscow, Vol. 3.
10. Captur, V.A., Hulyayev, P.S. and Kravchenko, K.D. (2012), "Basic principles of addressing the practical implementation of variable size network addresses to Ethernet Networks", *Radioelectronic and computer systems*, No. 1, pp. 51–54.
11. Lipaev, V.V. (2001), *Software quality assurance. Methods and standards*, Sinteg, Moscow.
12. Ruban, I., Kuchuk, H. and Kovalenko A. (2017), "Redistribution of base stations load in mobile communication networks", *Innovative technologies and scientific solutions for industries*, No 1 (1), P. 75–81, DOI : http://doi.org/10.30837/2522-9818.2017.1.075
13. Kosenko V. (2017) Mathematical model of optimal distribution of applied problems of safety-critical systems over the nodes of the information and telecommunication network, *Advanced Information Systems*, Vol. 1, No. 2, pp. 4-9, DOI: http://doi.org/10.20998/2522-9052.2017.2.01
14. Kuchuk, G., Kovalenko, A., Kharchenko, V. and Shamraev, A. (2017), "Resource-oriented approaches to implementation of traffic control technologies in safety-critical I&C systems", *Green IT Engineering: Components Network and Systems Implementation*, Springer International Publishing, Vol. 105, pp. 313–338.
15. Amin Salih M. and Potrus M.Y. (2015), "A Method for Compensation of Tcp Throughput Degrading During Movement Of Mobile Node", *ZANCO Journal of Pure and Applied Sciences*, Vol. 27, No 6, pp. 59–68.
16. Saravana, Balaji B,, Karthikeyan, N.K. and Raj Kumar, R.S., (2018), "Fuzzy service conceptual ontology system for cloud service recommendation", *Computers & Electrical Engineering*, Vol. 69, pp. 435–446, DOI: http://doi.org/10.1016/j.compeleceng.2016.09.013
17. Amin Salih, M., Yuvaraj, D., Sivaram, M. and Porkodi, V. (2018), "Detection And Removal Of Black Hole Attack In Mobile Ad Hoc Networks Using Grp Protocol", *International Journal of Advanced Research in Computer Science*, Vol. 9, No 6, pp. 1–6, doi: http://dx.doi.org/10.26483/ijarcs.v9i6.6335
18. Olifer, N.A. (2006), *Computer networks. Principles, technologies, protocols*, Piter, St. Petersburg.
19. Kaptur, V.A., Gulyaev, K.D., Kravchenko, P.S. and Yanina, O.O. (2011), "Estimation of efficiency of implementation of telecommunication technologies", *Otkrytye ynformatsyonnye y komp□yuternye yntehryrovannye tekhnolohyy,* Nats. aérokosm. un-ta ym. N.E. Zhukovskoho "KHAY.", No. 52, pp. 77–89.
20. Pleskach, V.L. and Zatonatskaya, T.G. (2011), *Information systems and technologies at enterprises*, Znannya, Kyiv.
21. Gomathi, B, Karthikeyan, N.K. and Saravana, Balaji B., (2018), "Epsilon-Fuzzy Dominance Sort Based Composite Discrete Artificial Bee Colony optimization for Multi-Objective Cloud Task Scheduling Problem", *International Journal of Business Intelligence and Data Mining*, Vol. 13, Issue 1-3, pp. 247-266, doi: http://doi.org/10.1504/IJBIDM.2018.088435
22. Dhivakar, B., Saravanan, S.V., Sivaram, M. and Krishnan R.A. (2012), "Statistical Score Calculation of Information Retrieval Systems using Data Fusion Technique", *Computer Science and Engineering*, Vol. 2, Issue 5, pp.43-45, doi: http://doi.org/10.5923/j.computer.20120205.01
23. Tannenbaum, E. (2007), *Computer Architecture*, Piter, St. Petersburg.
24. Tatarchuk, M.I. (2005), *Corporate Information Systems*, Textbook, Ministry of Education and Science of Ukraine, Kyiv National Economic University, Kyiv, 328 p.
25. Hart, M. J. (2005), *Windows System programming in the Windows environment*, Williams, Moscow.
26. Ivchenko, G.I., Kashtanov, V.A. and Kovalenko, I.N. (1982), *Queuing theory*, Vyssh. shkola, Moscow.
27. Douglas, E. (2003), "Principles, Protocols, and Structure", *TCP/IP networks*, Williams, Moscow, Vol. 1.
28. Korn, G.A (1968), *Handbook of mathematics for scientists and engineers. Definitions, theorems, formulas*, Nauka, Moscow.

29. Sah, J.J. and Malik, D.L.J. (2013), “Impact of DDOS attacks on cloud environment”, *International Journal of Research in Computer*, p. 276, available at: http://ijrcct.org/index.php/ojs/article/download/276/pdf.
30. Department of Special Telecommunication Systems and Information Security of the Security Service of Ukraine (1999), ND TZI 1.1-003-99,*Terminology in the field of information protection in computer systems from unauthorized access*, the Security Service of Ukraine, normative document, available at: https://tzi.com.ua/downloads/1.1-003-99.pdf.

Received (Надійшла) 16.12.2018
Accepted for publication (Прийнята до друку) 06.03.2019

Дослідження процесу обміну інформацією в локальних мережах установ за допомогою використання математичного апарату теорії масового обслуговування

О. А. Макогон, Р. Г. Мусаєв, О. О. Дичко, І. М. Криленко, Є. А. Думич

Предметом вивчення в статті є процес обміну інформацією в локальних мережах установ на комп’ютерах з різними технічними параметрами. **Метою дослідження** є аналіз клієнт-серверної взаємодії за допомогою сокетів, розробка програми для ефективного і безпечного спілкування в локальній мережі. **Задачі:** провести аналіз клієнт-серверної взаємодії у локальній мережі за допомогою використання сокетів, розробити програму спілкування в локальній мережі на комп’ютерах з різними технічними параметрами, використовуючи математичний апарат теорії масового обслуговування, провести дослідження процесу обміну інформацією у системі “клієнт – сервер” з метою підвищення його ефективності. Використовуваними є загальнонаукові та спеціальні **методи** наукового пізнання. Отримані такі **результати.** На сьогоднішній день парк обчислювальної техніки, що використовується в установах і офісах, має дуже велику дисперсію технічних параметрів та встановленого на комп’ютерах програмного забезпечення. Виходячи з цього, організація офісних локальних мереж вимагає вирішення науково-технічного завдання реалізації такої архітектури, яка б одночасно задовольняла вимогам якості, швидкості, зручності та рівню безпеки обміну інформації між комп’ютерами з різними Hardware и Software. Саме сокети доцільно застосовувати для протоколів, орієнтованих на встановлення зв'язку, що дозволяє одночасно обробляти запити від великої кількості користувачів. При цьому адміністратор в будь-який момент часу може вносити корективи в налаштування та здійснювати перерозподіл портів на серверах. Використання математичного апарату теорії масового обслуговування дало змогу провести дослідження процесу обміну інформацією у системі “клієнт – сервер” з метою підвищення його ефективності. Кожен сервер був описаний як замкнута одноканальна система масового обслуговування з необмеженим часом очікування. **Висновки.** Авторами розглядається реалізація архітектури “клієнт – сервер” в офісній локальній мережі, успішне вирішення якої дозволить працювати військовій установі оптимально і безперебійно. Авторами була розроблена програма спілкування в локальній мережі на комп’ютерах з різними технічними параметрами та проведене тестування програми на предмет виявлення помилок. Застосування теорії масового обслуговування дає змогу розв'язати завдання структурного та параметричного синтезу локальної мережі установи та відпрацювати методику виявлення розподілених атак на мережу на початковому етапі.

Ключові слова: архітектура “клієнт – сервер”; сокет; локальна мережа; теорія масового обслуговування; замовлення; одноканальна замкнута система масового обслуговування; інформаційна безпека.

Исследование процесса обмена информацией в локальных сетях учреждений с помощью использования математического аппарата теории массового обслуживания

Е. А. Макогон, Р. Г. Мусаев, А. А. Дычко, И. М. Крыленко, Е. А. Думич

Предметом изучения в статье является процесс обмена информацией в локальных сетях учреждений на компьютерах с различными техническими параметрами. **Целью исследования** является анализ клиент-серверного взаимодействия с помощью сокетов, разработка программы эффективного и безопасного общения в локальной сети. **Задачи:** провести анализ клиент-серверного взаимодействия в локальной сети посредством использования сокетов, разработать программу общения в локальной сети на компьютерах с различными техническими параметрами, используя математический аппарат теории массового обслуживания, провести исследования процесса обмена информацией в системе "клиент - сервер" с целью повышения его эффективности. Используемыми являются общенаучные и специальные **методы** научного познания. Получены следующие **результаты.** На сегодняшний день парк вычислительной техники, используемой в учреждениях и офисах, имеет очень большую дисперсию технических параметров и установленного на компьютерах программного обеспечения. Исходя из этого, организация офисных локальных сетей требует решения научно-технической задачи реализации такой архитектуры, которая бы одновременно удовлетворяла требованиям качества, скорости, удобства и уровню безопасности обмена информацией между компьютерами с разными Hardware и Software. Именно сокеты целесообразно применять для протоколов, ориентированных на установление связи, что позволяет одновременно обрабатывать запросы от большого количества пользователей. При этом администратор в любой момент времени может вносить коррективы в настройки и осуществлять перераспределение портов на серверах. Использование математического аппарата теории массового обслуживания позволило провести исследование процесса обмена информацией в системе "клиент - сервер" с целью повышения его эффективности. Каждый сервер был описан как замкнутая одноканальная система массового обслуживания с неограниченным временем ожидания. **Выводы.** Авторами рассматривается реализация архитектуры "клиент - сервер" в офисной локальной сети, успешное решение которой позволит военному учреждению работать оптимально и бесперебойно. Авторами была разработана программа общения в локальной сети на компьютерах с различными техническими параметрами и проведено тестирование программы на предмет выявления ошибок. Применение теории массового обслуживания позволяет решить задачу структурного и параметрического синтеза локальной сети учреждения и отработать методику выявления распределенных атак на сеть на начальном этапе.

Ключевые слова: архитектура "клиент - сервер"; сокет; локальная сеть; теория массового обслуживания; заказ; одноканальная замкнутая система массового обслуживания; информационная безопасность.