International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 12 (2020), pp. 5274-5282

Modeling a Transport-Level Telecommunication Management Service of Thermoelectric Systems Based on Petri Nets

Oleg R. Kuzichkin [1], Vladimir T. Eremenko [2], Gleb S. Vasilyev [1], Alexey V. Eremenko [2], Dmitry I. Surzhik [1], Sergey V. Eremenko [2]

[1] *Department of Information and robototechnic systems, Belgorod State University, Belgorod, 308015, Russia*.

[2] *Department of Information security, Oryol state University named after I. S. Turgenev, Oryol, 302026, Russia*.

Abstract

A promising direction for the organization of cooling systems in the agro-industrial complex is thermoelectric complexes built on the basis of thermoelectric modules. When developing and operating them, it is necessary to provide precise adjustment and correction of the operating modes of all distributed parts of the thermoelectric system for maintaining the microclimate in real time. This task can be solved by distributed telecommunication control systems. Based on the apparatus of Petri nets, a model of transport-level service for telecommunication management of thermoelectric systems has been developed. The interface of the transport layer and the scheme of the transport service data conversion module, including procedures for converting socket library data formats, have been developed. This ensures the integrity of the description of processes for establishing connections between environment objects and data exchange, as well as the compatibility of private models corresponding to subsystems with different levels of service services. A scheme for dispatching messages circulating in an ad-hoc environment has been developed, which operates on the principle of priority queue. This scheme allows to deliver urgent messages in the most efficient and cost-effective way.

Keywords: agro-industrial complex, thermoelectric system, automatic control system, telecommunication network, transport level, Petri net

I. INTRODUCTION

Currently, a promising direction for the organization of cooling systems in the agro-industrial complex is thermoelectric complexes built on the basis of thermoelectric modules [1,2]. They solve the problem of intensifying heat exchange to obtain the temperature of the cooled zones of agricultural production below the ambient temperature [3]. It should be noted that an important feature of the use of thermoelectric modules in the formation of microclimate in agricultural enterprises is the possibility of distributed formation of local microclimate features in the selected zones of objects. This makes it possible to significantly increase the energy efficiency of production with the use of distributed controlled thermoelectric elements [4]. However, it is necessary to take into account the need for precise adjustment and correction of operating modes of all distributed parts of the thermoelectric system for maintaining the microclimate in real time on the basis of distributed telecommunication control systems [5].

Considering the rapid development of telecommunication infrastructure, especially in recent years, it is difficult to identify priority technological solutions aimed at implementing network architectures in the control systems of the agro-industrial complex. This is especially evident in the open systems interconnection (*OSI*) model and the *Internet*. The *OSI* architecture was planned as the basis for global networks, which is why its protocols were so versatile and complex. The *Internet*, which was created at first as a combination of a small number of narrow-purpose military networks, turned out to be the largest data network in the world and continues to grow at an exponential rate, unexpectedly even for its creators. Its initially simple protocols were too tight for it, and it had to borrow a lot from the developed *OSI* [6].

From the pragmatic point of view of implementing network management technologies in the agro-industrial complex, it is better not to oppose the *Internet* and *OSI* protocols to each other, but to consider them as complementary. Therefore, developers began to create methods and means of interaction between both *IPS* protocol stacks (*Internet Protocol Suite*) and *OSI* in the same system or network [7]. Standards are being developed that use *OSI* application protocols over the *TCP/IP* infrastructure, and gateways are being defined between the two systems for passing messages of various formats. Several strategies for mutual transition and interaction in the communication environment of *Internet* and *OSI* applications have been developed: mixed and double protocol stacks, common application interfaces (*APIs*) and protocol translation methods; methods for direct conversion between both protocol stacks have been proposed [8].

It is advisable to connect various segments of the cooling equipment control networks at the transport level in order to ensure their effective interaction. From the point of view of protocol interaction, the most interesting is the transport service, the quality of which is characterized by its component services.

There is a distinction between transport services with or without a connection. Establishing a transport connection involves establishing a network connection and transmitting transport protocol data blocks (*TPDB*) over it, and is completed by similar processes when disconnecting. When transmitting *TPDB*, they are divided into separate fragments that are transmitted by network service data blocks [9].

Transport service is characterized by probabilistic and time parameters that allow us to formulate the necessary requirements for a transport connection and conclude that they are fully met. The transport service is special in point that it is connected to the services of the lower levels. The transport connection is characterized by the quality of service and passes through the following phases (stages): establishing a connection, transmitting data (maintaining a session), and ending the connection. The concept of "quality of service" refers to the connection parameters at all stages. We can select the following groups of parameters:

- parameters agreed upon during connection establishment;

- parameters whose values are selected independently, without agreement with the correspondent;

- parameters whose values are not selected, but are communicated to users [10].

These groups of transport-level service quality parameters fully characterize its time and probabilistic indicators. Time indicators determine the speed of data input and output and the time of data transfer. These include: connection delay, bandwidth, transit delay, and disconnect delay. Probabilistic indicators are related to the reliability of the connection. These include: the probability of failure to establish a connection, the coefficient of non-detection of errors, survivability, failure probability, and the probability of non-connection. Other parameters relate to special mechanisms at the transport level. The latter define different levels of data protection against unauthorized access (*UAA*), as well as the priority of message processing.

I.I Problem statement

Various information exchange protocols can be used in network technologies of telecommunication control systems for thermoelectric equipment of agricultural facilities. However, *TCP/IP* is the most suitable protocol. They are usually designed to solve certain tasks, so they are optimized and effective compared to the *OSI* protocols. However, they do not have a standard set of application service elements or design blocks that allow us to build reliable distributed control systems. Therefore, the task of modeling the transport layer service in the *TCP/IP* protocol stack is relevant, taking into account the theoretical achievements of the reference model and the description of information exchange processes [11].

The choice of the level of interaction between *TCP/IP* protocols depends largely on the choice of the network programming method. Currently the following programming methods are most widely used: sockets, *RPC*, and *TLI*. All of them are used in *UNIX*-like programming environments. However, in addition to this environment, the vast majority of users, including in telecommunication environments of enterprises, work in a *Windows* environment, in which only the socket method is suitable for modeling a low-level service for building distributed transport systems [12]. In this case, the socket is characterized as a special object created for sending and receiving data over the network. Note that the term "object" in this case does not mean an object in terms of object-oriented programming, but some entity whose internal structure is hidden from us, so we can only deal with this entity as a single and indivisible (atomic) object. This object is created inside the socket library, and the programmer using this library gets a unique number (handle) for this socket. In order for two programs to communicate with each other over the network, each of them must create a socket. Each socket has two main characteristics: the protocol and the address to which it is bound. The protocol is set when creating the socket and cannot be changed later. The socket address is set later, but always before data is sent through the socket. In some cases, binding a socket to an address may be implicit. The format of the socket address is determined by the specific protocol. In particular, for the *TCP* and *UDP* protocols, the address consists of the *IP* address of the network interface and the port number. Protocols in the *TCP/IP* stack can be used to identify various applications or ports. The port number associated with the application and the *IP* address of the host where the application runs uniquely identify the application on the network. The network address is combined with the port number and forms a socket. The socket interface allows interaction between processes, regardless of whether they are running on the same object or on different ones. In addition, the interface provides a single set of functions for working with different protocol stacks. Sockets provide two-way point-to-point communication between two processes. They are the main components of intersystem and inter-process communication. Each socket represents a connection endpoint that can have a name associated with it. It also has a specific type, and one or more associated processes.

These circumstances, the universality of the programming method and its use on *UNIX* and *Windows* platforms, led to the choice of sockets for modeling the transport layer.

Given the proximity of the socket library interface to the *TLI* transport layer interface, the negotiation of the *TCP/IP* protocol suite with the upper layers of the *OSI* will consist in

developing an additional socket library interface that allows working with data in the *TLI* transport layer format. When switching to the *TLI* interface, one will not need to rework the existing software.

Any reconfigurable complex system is very similar in nature to open systems, whose users themselves perform some of the functions of their creators. The main task here is to develop methods for modeling transport-level service in telecommunication environments of enterprises using Petri nets, so that the potential consumer can design the necessary configuration, structure of the environment, the degree of duplication of nodes and communication channels, distribute the functional load and control between the nodes of the network in various operation modes of the object [13,14].

I.II Service diagram for modeled levels

For the developed service model of a network of telecommunication control systems for thermoelectric equipment of agricultural facilities, the most appropriate one is a scheme of hierarchical inheritance of multilevel service functions, well described by the device of Petri nets (Fig. 1).

Petri net structures are a formal means of describing complex processes and are intended for building and investigating models of parallel computing systems [15].

The Petri net is represented as a directed graph with two types of vertices: positions *P* and transitions *t*, where vertices of different types are connected by arcs or arrows. Placemarks are placed in the network position that show the dynamics of the process as they move. Placement of tags (chips) in positions is called Petri net marking [16].

When setting a network, its initial marking is always specified. It changes as a result of performing transitions due to external events. A transition is triggered if all its input positions contain placemarks (or at least one marker) and an event occurs. When a transition is triggered, a chip is removed from each of its input positions, and a chip is added to each output position.

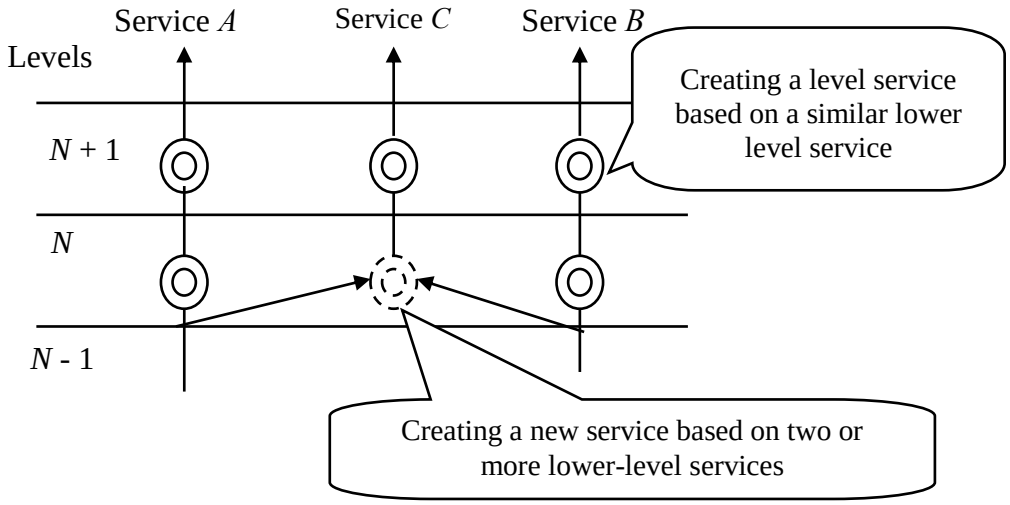

Figure 1. Service diagram of the developed model

Arrows from transition to positions determine the transition of the marker to positions when the transition is triggered, which corresponds to the transfer of control from one procedure to another. These positions are called outputs for this transition [17]. Obviously, the output position of a single transition can be the input position for one or more subsequent transitions. In the presented scheme of information exchange (Fig. 2), positions *P*1, *P*5 that are not output positions of any transitions are called external input positions. The *P*9 position that is not the input position of any transitions is called the external output position. External input positions determine the conditions for the start of the protocol (or its block), and the external output position (there may be more than one) – the end of the protocol. All internal positions show the relationship of individual procedures to each other.

During the operation of the enterprise's telecommunication environment, information is constantly exchanged between the nodes of this system. The duplicate positions *P*2 and *P*6 simulate connecting nodes to the system, the duplicate positions *P*3 and *P*7 show the interaction between nodes through the transitions t_5 and t_6, and the positions *P*4 and *P*8 indicate disconnecting nodes from the system.

The gravitation *V* of two procedures P_i and P_j to each other refers to the presence of control transmission signals to each other and the amount of data transmitted from one procedure to another.

When decomposing the protocol into protocol blocks (*PB*), we can use varieties of Petri nets in the form of loaded and probabilistic networks, in which the arrows to the transitions indicate the load or probability of transition from one transition to another [20].

Users of application-level services are application processes generated by application programs. Information is exchanged through requests and responses. The application layer processes incoming data and prepares it for transmission to the application process or lower level. Data flows through the layer stack to the transport layer, which is represented by a set of functions and services of *TCP/IP* protocol sockets. The transport layer is responsible for direct data transmission. The upper levels provide a common service for the functioning of the environment based on an organized transport connection.

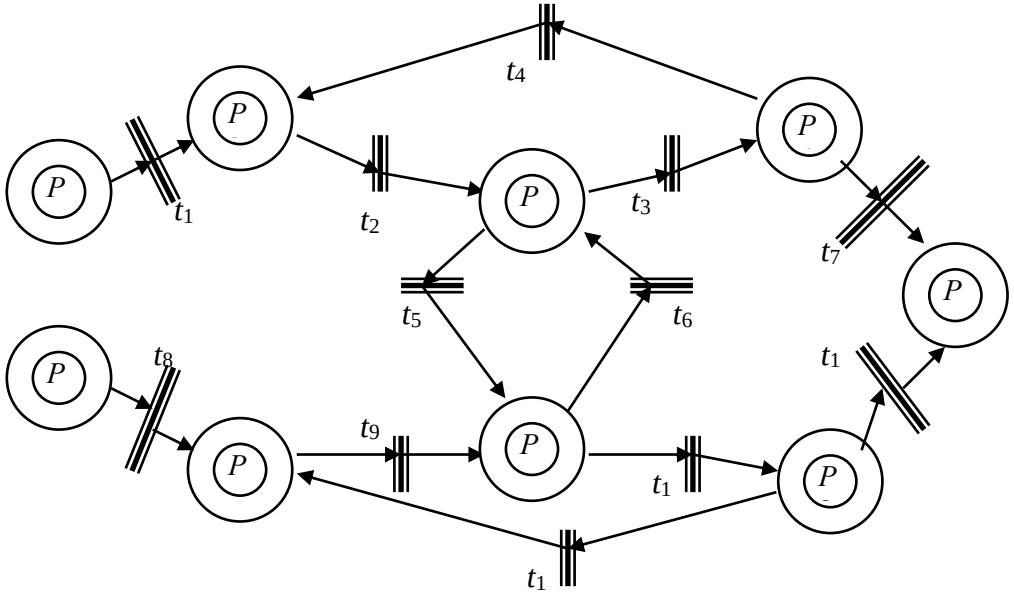

Figure 2. Diagram of information exchange in an ad-hoc network environment represented by a Petri net

The client-server architecture is the dominant architecture for an enterprise's telecommunication environment. In it, client applications request services from the server process. The client and server require a standard set of agreements before starting work–a protocol that is mandatory for all exchange participants. The protocol can be symmetric or asymmetric. In a symmetrical protocol, correspondents are equal. In asymmetric – one side is the main, the other – the subordinate. An example of a symmetric protocol is *TELNET*, which is used in the *Internet* to emulate a remote terminal. An example of an asymmetric protocol is *FTP*. Regardless of the protocol type, there are "process-server" and "process-client" when accessing the service.

The process server is in request waiting mode at a well-known address. It is inactive in this mode. After receiving the request, the process server is activated and serves the client by performing certain actions.

The enterprise's distributed telecommunication environment contains objects that interact with each other through established application associations. Depending on the tasks to be solved, each object can set an arbitrary number of application associations. The object management program is based on the service being developed.

Interaction of objects of the enterprise's telecommunication environment, shown in Fig. 3, is based on Petri nets apparatus.

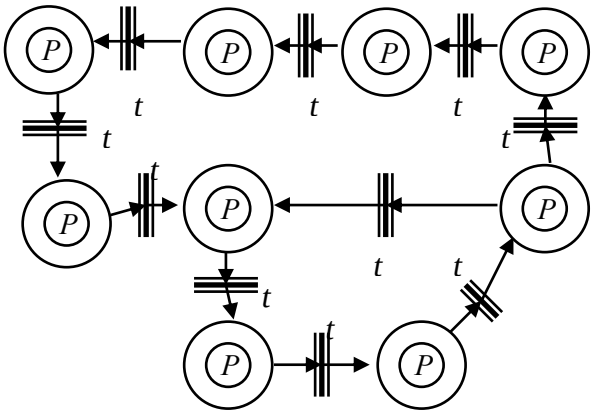

Figure 3. Objects' interaction model in the telecommunication network of the thermoelectric equipment control system

Positions have the following purposes:

- *P*1 defines receiving the initialization command (enabling the system);
- *P*2 prepares the variables and data structures that are needed to work;
- *P*3 initializes ports for reading connection requests and

establishes application associations necessary for starting the system;

*P*4 accepts connection requests on ports and application messages on installed application associations, and performs primary data processing;

*P*5 performs final data processing and analysis of the current state of a part of the environment or the environment as a whole;

*P*6 prepares to send a message to the environment's recipients via installed application associations or commands to connected industrial equipment;

*P*7 sends ready data to recipients or commands to hardware;

*P*8 destroys variables and data structures;

*P*9 terminates the program, all side processes, and closes all application associations.

Representing the environment of an ad-hoc network as a Petri net provides a deeper insight into the relationship of procedures. As a result of decomposition of the interconnected system of remote objects based on Petri nets, a more economical composite protocol model of the system is obtained than when using the model of an algorithmic automaton [18, 19].

I.III Developing the transport layer interface

The simulation of the telecommunication environment service of control systems for thermoelectric equipment of agricultural facilities is based on the transport layer with the corresponding services provided by the socket library or *TLI*. We need a specific interface to work with them. The most serious difference between the socket library interfaces and the *TLI* transport layer is the difference in the service delivery data formats. They relate to the basic concepts of a socket and a transport endpoint. The task is to organize a new interface data type – an endpoint access to services and develop a data conversion module for the transport layer. This type of data can work with both of the original formats and, depending on the transport layer environment used, generate the appropriate data for providing the transport service. The operation diagram of the data conversion module is shown in Fig. 4. Its presence is a necessary condition for inter-level interaction.

Transport service primitives are implemented by unambiguously mapping the socket library or *TLI* library to procedures. Implementing a service for connections to a remote object means sending a switching request over the network (a client function) and setting a pre-set port in the event that the remote process also sent a switching request (a server function). If a connection is attempted simultaneously from both sides, two connections may be established, one of which may be closed by mutual agreement of the parties. The priorities of the parties for the right to close the second connection must be hard-defined and passed for comparison with the service primitive of the *T-CONNECT* connection request.

During the operation of the telecommunication environment, various types of industrial equipment interact with each other, and various service subsets are used for this purpose. At the session level, new functions are being added: managing data exchange markers, data synchronization, and activity.

Therefore, when implementing a service level, new data structures are introduced that include:

– marker flags showing whether or not markers are present on this side of the session connection;
– activity flag;
– current number of the sync point;
– link to the data corresponding to the connection state at the last confirmed big sync to restore the lost session connection.

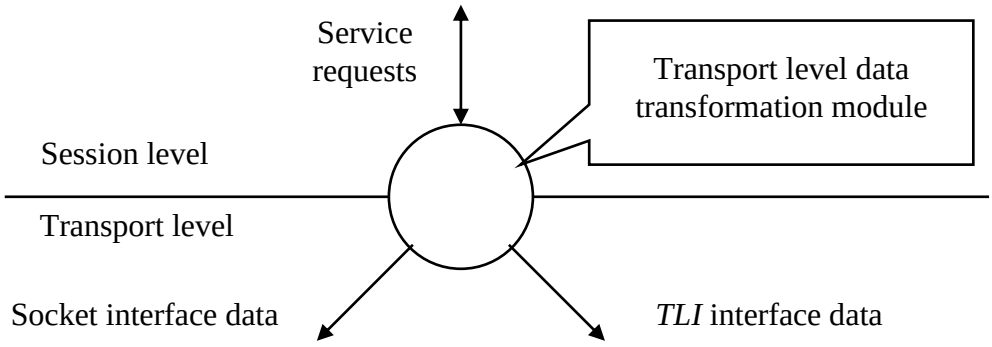

Figure 4. Operation diagram of the data conversion module of transport service

When conducting a session connection, performing the *S-CONNECT* service, the parties agree on the following parameters:

1. A set of functional groups required for the connection to function.
2. Initial placements of markers corresponding to the selected functional groups.
3. The initial value of the synchronization point (if the activity management group was not included in the list of required ones).

The algorithm for performing the *S-CONNECT* service on the part of the connection initiator (client) is shown in Fig. 5.

The *S-CONNECT* session level service is mapped to the following transport level services: *T-CONNECT*, *T-DISCONNECT*, and *T-DATE*. All session-level services are displayed in the same way.

Data format conversion operations are performed at the representative level. The functions of the representative level

are distributed among the procedures for forming data structures at other levels.

These functions consist in converting data from a specific computer format to a standard network format to preserve the byte order. The procedures for converting data formats are shown in Table 1.

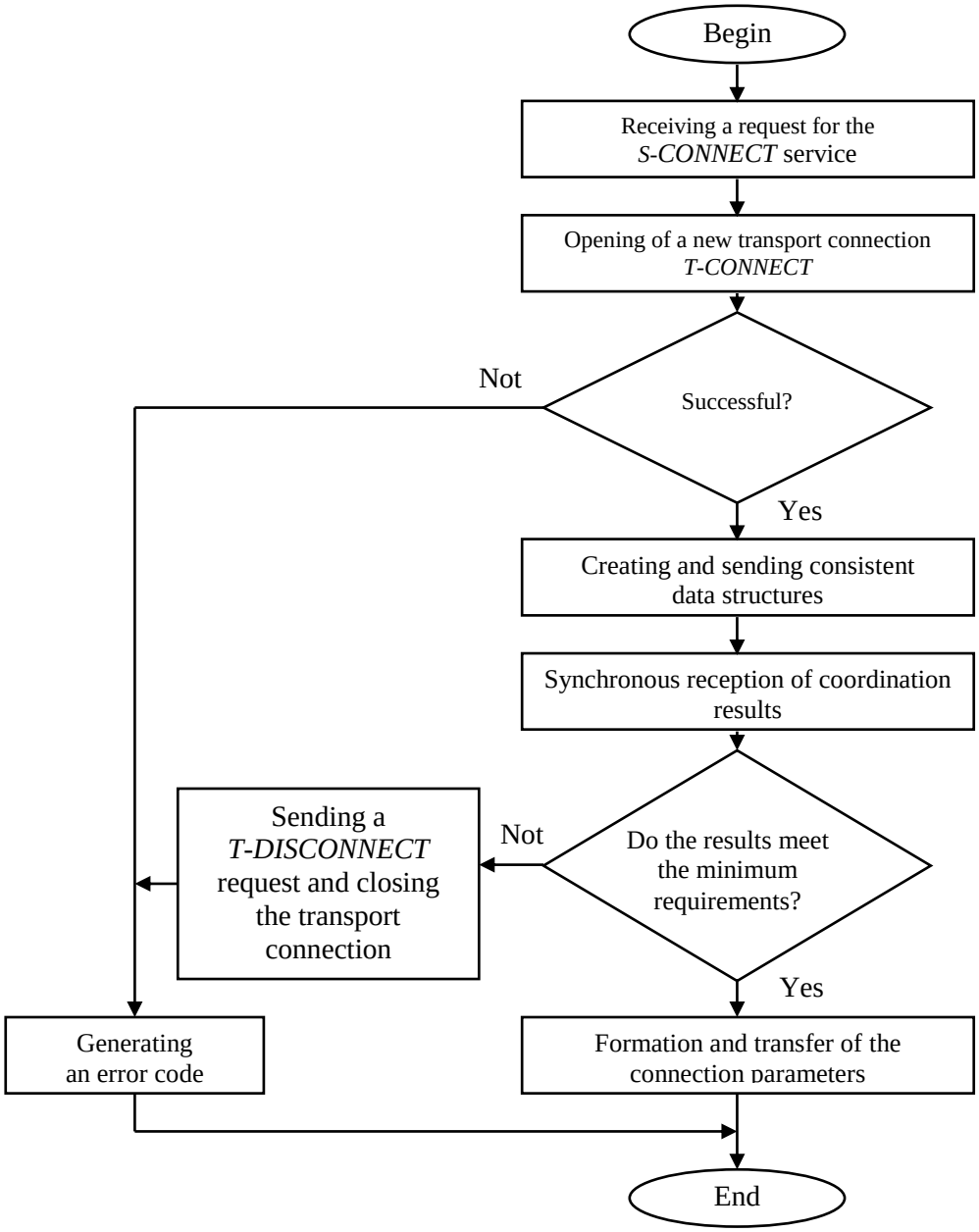

Figure 5. *S-CONNECT* service execution algorithm

Table 1. Procedures for converting data formats of the library of sockets

Procedure	**Description**
bcmp (*s*1, *s*2, *n*)	Byte string comparison: same → 0, different → 1
bcopy (*s*1, *s*2, *n*)	Copy *n* bytes from *s*1 to *s*2
bzero (*base*, *n*)	Filling *n* bytes with zeros, starting from the beginning
htonl (*val*)	*host-to-network-long*, converting a 32-bit byte order from machine to network
htons (*val*)	*host-to-network-short*, converting the byte order of a 16-bit number from machine to network
ntohl (*val*)	*network-to-host-long*, converting a 32-bit byte order from network to machine
ntohs (*val*)	*network-to-host-short*, converting the byte order of a 16-bit number from network to machine

I.IV Development of an information exchange model of the enterprise's telecommunication environment

The scheme of an information exchange of the enterprise's telecommunication environment is shown in Fig. 6.

The main components of the object are:

1. Data about the state of the object and the part of the network managed by it.

2. Message manager, which organizes the reception of data from installed application associations, transfer them to the appropriate processing procedures and send them to network partners. To ensure the safety of incoming and outgoing messages, the corresponding data storage stacks are used, working on the principle of a queue (first-in – first-served).

3. Message processing procedures, each of which corresponds to a specific message type and has its own priority.

4. List of application associations installed by the object.

5. The main program loop, which provides direct control of the manager, communications, data processing, state estimation of the object and a controlled part of the telecommunication environment of the enterprise. Based on the received data, decisions about further actions are made and control signals are generated.

The execution of the main program cycle is described using a Petri net (Fig. 8).

Network positions have the following purpose:

*P*1: preparing data for a new cycle of the management program;

*P*2: receiving the next message with the highest priority from the outgoing message stack;

*P*3: passing the prepared message through the appropriate application association and deleting it from the outgoing message stack;

*P*4: processing the state of installed application associations; receiving messages and transmitting them to the incoming message stack in accordance with the priority;

*P*5: selection of the next message with the highest priority from the incoming message stack;

*P*6: passing a message to the appropriate processing procedure and deleting it from the stack;

*P*7: assessment of the state of the object and part of the environment as a whole;

*P*8: making decisions and sending necessary messages to partners on the enterprise's telecommunication network; organizing new application associations.

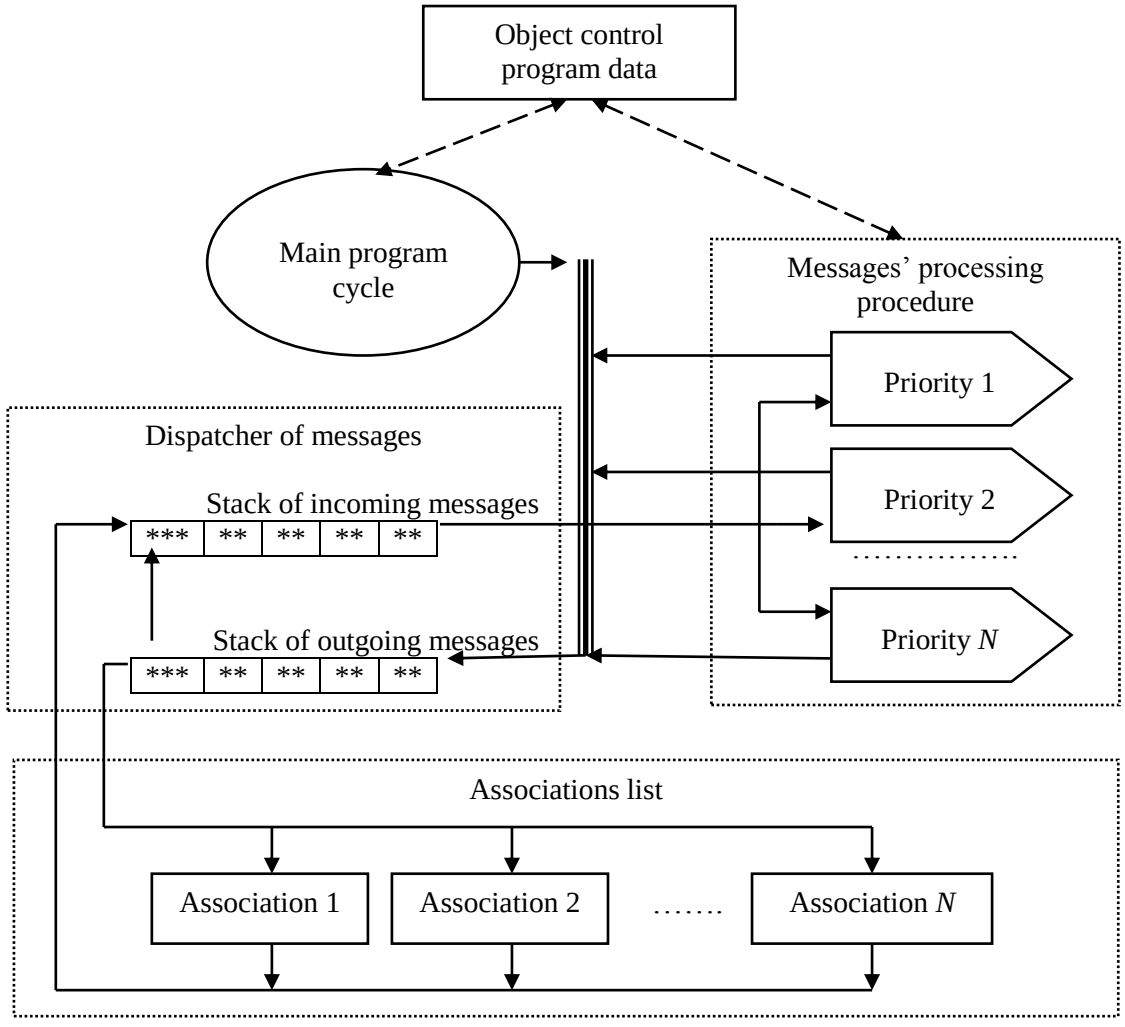

Figure 6. Diagram of information exchange processes in an ad-hoc network environment

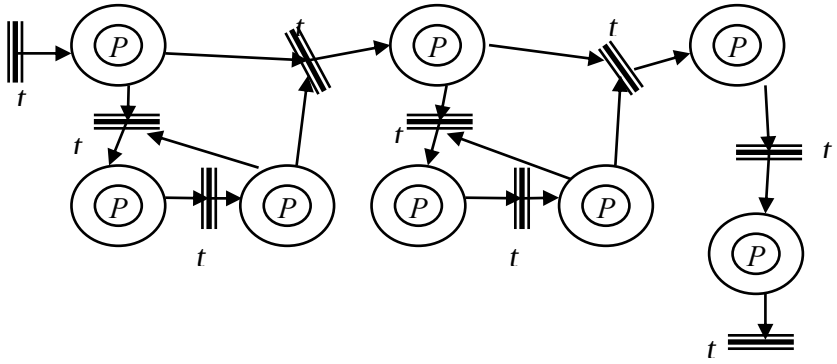

Figure 7. Main program cycle controlling the object of an enterprise's telecommunication environment

Transitions *t* have the following trigger conditions:

t_3: passing control to the object management program;

t_2: there is a sending queue in the outgoing message stack;

t_3: another message is received from the outgoing message stack;

t_4: the outgoing message stack is empty;

t_5: there is at least one message in the incoming message stack;

t_6: the next message with the highest priority is received from the incoming message stack;

t_7: the incoming message stack is empty;

t_8: the state of the object and the environment as a whole is assessed;

t_9: messages were sent to partners in the enterprise network in accordance with the decisions taken; actions were taken to organize new application associations.

The use of the Petri net apparatus for the shown algorithm allows us to visually assess the depth of relationships between procedures. The developed model of information exchange in an ad-hoc network environment most fully describes the processes that occur when managing an object and it allows developers to design the environment depending on the tasks assigned to the object and taking into account all the source data.

One of the most important parts of the main cycle of the object's management program is processing the state of installed application associations, namely, receiving and analyzing messages from partners. With the use of a group of services for predicting changes transmitted over the environment of an ad-hoc network, this procedure becomes more complicated. A list of data structures registered for this service group is added to the data describing the application association.

II. CONCLUSION

Thus, it is most appropriate to build a service for telecommunication management of thermoelectric equipment of agricultural facilities using socket-based network programming methods, taking into account the hierarchical inheritance of services from the lower to the upper levels of information exchange. To model the service, we used a hierarchical Petri net apparatus that allowed us to consider the behavior of an object taking into account the parallelism of network processes in the client-server architecture. When building an ad-hoc network environment, it is advisable to consider all objects conditionally equal. However, the application layer should hide most of the asymmetric processes. The most difficult part of the basic core of application services to implement is the process of establishing an application association because of the many parameters that need to be agreed upon when providing this service.

The possibility of developing and integrating a transport-level service model for *TCP/IP* protocols in ad-hoc network environments using queuing theory and graph theory is substantiated. This ensures the integrity of the description of processes for establishing connections between environment objects and data exchange, as well as the compatibility of private models corresponding to subsystems with different service levels. A scheme for dispatching messages circulating in an ad-hoc environment has been developed, which operates on the principle of priority queue. This scheme allows to deliver urgent messages in the most efficient and cost-effective way.

ACKNOWLEDGMENTS

The article was prepared as part of the state task "Research and development of complex energy-saving and thermoelectric regenerative systems" application number 2019-1497, subject number FZWG-2020 -0034.

REFERENCES

[1] Bulat LP. Thermoelectric cooling. Saint Petersburg: Spbgunipt. 2002: 147 p.

[2] Anatychuk LI. Current state and some prospects of thermoelectricity. Thermoelectricity. 2007;2:7-20.

[3] Filin SO, Zakshevsky B. Modern state and prospects of development and production of stationary thermoelectric refrigerators. Thermoelectricity. 2008;2:74-88.

[4] Elizarov IA, Martemyanov YuF, Skhirtladze AG, Frolov SV. Technical means of automation. Software and technical complexes and controllers: Textbook. Moscow: "publishing house Mashinostroenie-1". 2004: 180 p.

[5] Kuzichkin OR, Eremenko VT, Vasilyev GS, Eremenko AV, Fisun AP, Surzhik DI. Organization and control of protocols for modular structures for collecting and processing geoecological monitoring data. International scientific and technical journal. 2020;29(11s):228-237.

[6] Scherbo VK. Standards of computer networks. The interconnection of networks. Reference book. - Moscow: "Kudits-Obraz". 2000: 268 p.

[7] Chappel L, Tittel E. TCP / IP. Training course / Translation from English by Yu. Gorokhovsky. - Saint Petersburg: BVC-Petersburg. 2003: P. 29. - 976 p.

[8] Krovchik A, Nomanlagari VK, et al. NET network programming for professionals. Trans. from eng. V. Streltsov. - M.: Lori. 2005: 400 p.

[9] The Intelligent Transport Layer - Available at: http://www.zeromq.org (accessed 23 September 2014).

[10] The Protocols of the Internet. Encyclopedia. Hotline-Telecom", Moscow. 2001: 1100 p.

[11] Hoar Ch. Interacting sequential processes, Moscow: Mir. 1989: -264 p.

[12] Eremenko VT. the Concept of detecting and correcting logical errors in the implementation of security Protocol profiles. Telecommunications. 2003;8:30-35.

[13] Eremenko VT, Konstantinov IS. Modeling of processes for analyzing implementations of information exchange protocols for solving problems of describing their static and dynamic interaction. Bulletin of computer and information technologies. 2004;4G: 11-15.

[14] Kuzichkin OR, Grecheneva AV, Eremenko VT, Loginov IV, Eremenko AV, Obozov AA. Application of the multi criterial ranking method for the integration of a regional gis into a geotechnical monitoring system. Journal of advanced research in dynamical and control systems. Том: 11. Номер: 8 SpecialIssue Год: 2019 Страницы: 384-394.

[15] Marakhovsky VB, Rosenblum LYa, Yakovlev AV. Modeling of parallel processes. Petri net. A course for system architects, programmers, system analysts, and designers of complex control systems. - Saint Petersburg: Professional literature, It-Preparation. 2014: - 400 p.

[16] Wenjing Li, Songzhao Li, Jianbo Lu. Research on Petri Net System Parallel Subnet Partitioning Completeness Theory and Algorithm.Wuhan University Journal of Natural Sciences. 2019;24:pp. 205.

[17] Eremenko VT. Eremenko AV, Ozarenko OV. Models and algorithms for analyzing logical errors in information exchange protocols. News of higher educational institutions. Volga region. Technical science.- Penza: Penza state University Press. 2006;6: 205-214.

[18] Mahulea C, Seatzu C, Cabasino MP, Silva M. Fault diagnosis of discrete-event systems using continuous Petri nets. IEEE Trans. Syst. Man Cybern. A Syst. Humans. 2012;42(4):970-984.

[19] Nikolaevich Makarov A, Vladimirovna Maksyutina E, Azretovich Hubiev K, Damirovich Gimadeev A. Global and national food security issues in the context of land and water resources. Caspian Journal of Environmental Sciences. 2020 Dec 1;18(5):467-72.

[20] Kuzichkin OR, Vasilyev GS, Grecheneva AV, Loginov IV, Eremenko VT, Eremenko SV. Uncertainty of the implementation time of geodynamic monitoring system in multi-criteria ranking of alternatives. Indonesian Journal of Electrical Engineering and Computer Science. 2019;17(3):1249-1257.