HIERARCHICAL COMPOSITION OF PROTOCOLS

N. A. Anisimov

Avtomatika i Vychislitel'naya Tekhnika,
Vol. 24, No. 1, pp. 3-10, 1990

UDC 681.325

The author deals with the formalization of the concept of protocol hierarchy within the framework of the theory of Petri nets. A formal definition of the concept of entity and of its points of interaction is introduced. An entity is defined as a Petri net, while interaction points are defined as the set of its markings. The operation of composition of entities is presented. Rules are given for formalizing operations with parameters and variables in the composition process. A description is offered of the structure of protocol entities as a composition of entities describing the protocol and the service furnished and required. The concepts introduced are illustrated by the procedures of Recommendation X.25.3 and of the class-0 transport protocol. The concept of timer and of procedures for its use is formalized.

A central problem in developing software for computer networks is the development of the set of protocols making up the multilayer hierarchical structure. Recently, there has been universal interest in changing over gradually from protocols directly to rules for configuring them into protocol sets [1,2]. This is a consequence of the fact that work on specifying the protocols of most layers has been completed, as well as of the now-prominent problems of compatibility of standards of different levels (see, e.g., [3]). Entire protocol sets are now becoming the object of standardization [1]. All this raises problems of formalizing the concept of protocol hierarchy and of developing rules for their hierarchical composition to the status of pressing problems in protocol theory.

Whereas there have been many studies dealing with protocol specification and analysis, problems of formalization of protocol hierarchies have as yet received much less attention.

At the same time, such important characteristics as compatibility of implementations, complexity and efficiency, and correctness of operation depend on the mode of hierarchical composition of protocols. Earlier studies along these lines, which employed formal grammars [4] and Petri nets [5], made attempts to establish a direct correspondence between the execution of protocol commands of different levels. Paper [6] proposed a simple hierarchical composition rule based on the operation of superposition of the algebra of regular Petri nets [7]. In practice, however, in the case of complex connections between layers, these approaches become unsuitable, since they do not rely on the Basic Reference Model of Open Systems Interconnection [8], which has a well-developed conceptual system for representing protocol hierarchies on an informal level. Later studies along these lines relied on the concept of protocol entity [9]. The notion of protocol entity was formalized on both the level of formal models [10-12] and on the level of protocol specification language. An example of the latter approach is provided by the languages OCA [13] and PDL [14], whose constructs directly contain terms of the Reference Model.

This paper deals with the formalization of the concept of protocol hierarchy on the formal-model level. We introduce a new (and, in our opinion, felicitous) interpretation of the concept of protocol entity in terms of Petri nets. Our work is a further development of the formal model of [15], intended for protocol development and description, and constitutes a more complete and more highly elaborated version of [16].

INTRODUCTION TO PROTOCOL HIERARCHY

A hierarchical protocol system can be represented as a single unified protocol. Figure 1 shows a schematic depiction of a two-layer system, containing N- and (N + 1)-protocols and (N − 1)-service. Here the N- and (N + 1)-layers can be represented as one layer in which the N- and (N + 1)-entities form a single entity, while the N- and (N + 1)-protocols form a single H-protocol. The formats of the H-protocol commands have an embedded structure, and comprise the N-protocol command formats, which can contain commands of the (N + 1)-protocol.

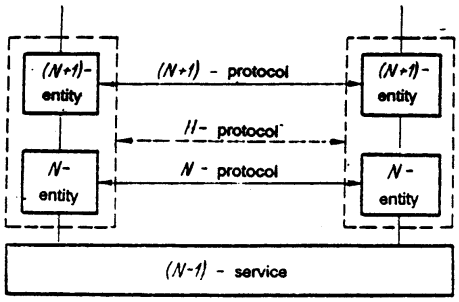

Fig. 1. Protocol hierarchy.

The sequence of execution of H-protocol commands satisfies the N- and (N + 1)-protocols. As compared to the N- and (N + 1)-protocols, however, the H-protocol contains considerable additional information (e.g., information on the N-protocol commands that contain (N + 1)-protocol commands; how the identifiers and parameters of the N- and (N + 1)-protocol commands are related; how the execution of N-protocol commands affects that of (N + 1)-protocol commands; and so forth). All this information should be contained in hierarchical composition rules. It should be noted that the information on hierarchical dependence is global rather than local in nature, and should also be an object of standardization.

According to the Reference Model of Open Systems Interconnection, information on hierarchical protocol dependence is contained in protocol entities. N-entities establish the possible sequences of execution of N- and (N — 1)-service primitives, as well as of N-protocols. Interaction of entities within one system is effected via service access points, at which service primitives are executed. Consequently, formalization of hierarchical protocol dependences involves formalization of protocol entities and of their interaction rules.

FORMAL DEFINITION OF ENTITY

We will introduce the basic concept of entity on the level of control structure. This suffices to demonstrate the basic ideas of the proposed approach. First we will give some necessary definitions from the theory of Petri nets.

Definition 1. A Petri net will refer to a combination N = <P, T, I, O, M_0>, where $P = \{p_1, p_2, \ldots, p_n\}$ is the set of places; $T = \{t_1, t_2, \ldots, t_m\}$ is the set of transitions; $I: P \times T \to \{0, 1\}$ and $O: P \times T \to \{0, 1\}$ are the input and output incidence functions; and $M_0: P \to \{0, 1, 2 \ldots\}$ is the initial marking function.

Assume that Σ is some finite set of names, called an alphabet. Symbol $\tau \notin \Sigma$ will denote some invisible action.

Definition 2. A marking of a Petri net refers to a combination $\alpha = \langle \Sigma_\alpha, \sigma_\alpha \rangle$, where Σ_α is some alphabet, while σ_α: $T \to \Sigma_\alpha \cup \{\tau\}$ is a marking function.

In what follows, we will use letters α, β, γ, ... to denote markings. A marking function marks network transitions by names from Σ; different transitions may have the same name, while some transitions may be unmarked (i.e., marked by τ). Unmarked transitions are usually used to describe internal (invisible) actions.

Definition 3. Petri net N together with marker α is called a marked Petri net: $N_\alpha = \langle N, \alpha \rangle$.

Definition 4. An entity refers to a collection E = <N, Γ>, where ν is a Petri net, called the structure of the entity, while $\Gamma = \{\alpha_1, \alpha_2, \ldots \alpha_k\}$ is the set of markers, called interaction points.

Thus, an entity is a Petri net. An entity can be interacted with (and hence its operation observed) only via interaction points. It is obvious that the entity will manifest itself in different ways at different interaction points. A transition marked by a τ at some point is assumed to be invisible, and interaction via it is impossible. A given transition can be seen from different points (to be sure, under different names). It is also possible to have a case in which a transition is seen from one point and not from another. A transition may be invisible altogether. If an entity does not have points of interaction at all, i.e., $\Gamma = \varnothing$, then obviously it cannot interact with other entities, i.e., it cannot manifest itself in the environment.

An entity will be represented schematically as a rectangle with lines, representing interaction points, that depart from it (Fig. 2a).

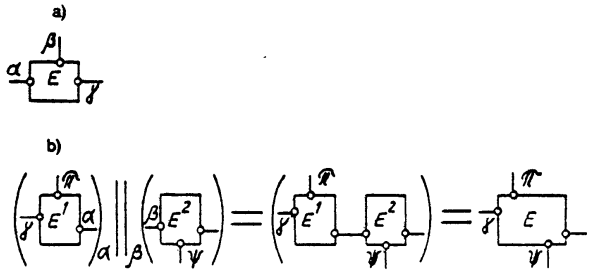

Fig. 2. Schematic representation of entity (a) and of composition of entities (b).

COMPOSITION RULES FOR ENTITIES

Initially we will introduce the two ancillary operations of union and composition over networks.

Assume that we are given networks N^1 and N^2. In what follows, we will assume that the networks have dissimilar sets of transitions and places, i.e., $P^1 \cap P^2 = \varnothing$ and $T^1 \cap T^2 = \varnothing$.

The union of networks N^1 and N^2 refers to network $N = N^1 \cup N^2 = \langle P^1 \cup P^2, T^1 \cup T^2, I^1 \cup I^2, O^1 \cup O^2, M_0^1 \cup M_0^2 \rangle$, i.e., two different networks are regarded as one network.

Assume that $X = \{x_1, x_2, \ldots, x_k\}$ and $Y = \{y_1, y_2, \ldots, y_l\}$ are two subsets of transitions of network N: $X \subseteq T$ and $Y \subseteq T$; the expression $\mu(N, X \otimes Y)$ denotes the operation of merging of transitions, which is formally defined in a manner absolutely analogous to the operation of merging of places [7], and involves the following. Each transition x_i is copied together with the incident arcs, for a total of l copies $x_i^1, x_i^2, \ldots, x_i^l$, while each transition y_j is copied as k copies $y_j^1, y_j^2, \ldots, y_j^k$. Then each pair (x_i^j, y_j^i) is replaced by a new transition $x_i \cup y_i$ together with the incident arcs: $I(x_i \cup y_j) = I(x_i) \cup I(y_j)$; $O(x_i \cup y_j) = O(x_i) \cup O(y_j)$.

Assume that for networks N^1 and N^2 we are given markers α and β, defined on the same alphabet: $\alpha = \langle \Sigma, \tau_\alpha \rangle$, $\beta = \langle \Sigma, \tau_\beta \rangle$. Then the composition of networks N^1 and N^2 in relation to markers α and β, denoted henceforth as $(N^1{}_\alpha \| _\beta N^2)$, unites these networks, merging transitions with the same names. This operation can be defined more rigorously as follows. Let $a \in \Sigma$. The set of transitions with the same name will be denoted as $T_a^i = \{t / \sigma(t) = a,\ t \in T^i\}$, $i = 1, 2$. Then $N = (N^1{}_\alpha \| _\beta N^2) = \mu(N^1 \cup N^2, Z)$, where $Z = \bigcup_{a \in \Sigma} T_a^1 \otimes T_a^2$.

It should be noted that this operation merges only visible transitions. Assume that network N^1 has marker γ. We set up a continuation of marker γ, denoted by γ^*, for network $(N^1{}_\alpha \| _\beta N^2)$, in natural fashion: $\sigma^*_\gamma(t) = \sigma_\gamma(t)$, $t \in T^1$; $\sigma^*_\gamma(t_i \cup t_j) = \sigma_\gamma(t_i)$; $\sigma^*_\gamma(t) = \tau$, $t \in T^2$. We also continue network Γ: $\Gamma^* = \{\alpha^*_1, \alpha^*_2, \ldots\}$.

Assume that $E^1 = \langle N^1, \Gamma^1 \rangle$ and $E^2 = \langle N^2, \Gamma^2 \rangle$ are entities and that their interaction points $\alpha \in \Gamma^1$ and $\beta \in \Gamma^2$ have a common alphabet $\Sigma_\alpha = \Sigma_\beta$.

Definition 5. A composition of entities E^1 and E^2 relative to interaction points α and β refers to entity $E = \langle N, \Gamma \rangle$ for which $N = (N^1{}_\alpha \| _\beta N^2)$, $\Gamma = (\Gamma^1 \setminus \{\alpha\})^* \cup (\Gamma^2 \setminus \{\beta\})^*$.

In other words, a composition of entities relative to interaction points involves the composition of their networks relative to the corresponding markers, after which these markers are excluded from the definition of the entity. Figure 2b offers a schematic representation of the execution of the operation of composition. Figure 3 shows an example of composition of entities consisting of a single transition. Here each entity has two interaction points, and each transition is marked by two names. The composition is relative to points with the alphabet.

An elementary interaction in composition of entities involves merging of transitions. This mechanism is well known as the "rendezvous" interaction scheme, and is widely employed in various formalisms [17-20]. The composition operation is quite similar to parallel composition in CCS [19] and CSP [20], and is essentially an expression of it in terms of marked Petri nets. In other words, a structural analogy has been provided for parallel composition of processes.

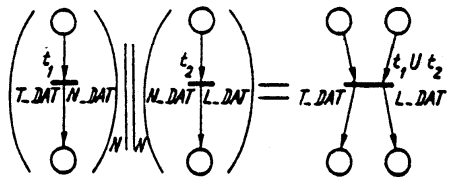

Fig. 3. Example of composition of objectives.

SPECIFICATION OF ENTITIES

In accordance with the definition of entity introduced, the description should involve specification of a Petri net that describes the structure of the entity, and of a set of markers. Papers [6,15] propose a method of algebraic specification of entities, although admittedly with one interaction point, specified only on a set of protocol commands. At the same time, practical specification of protocol entities and their procedures makes simultaneous use of multiple sets of commands (protocol commands, commands related to services offered and required, timer-interaction commands, and so forth). In what follows, the algebraic method of protocol description is generalized to the case of description of entities.

Assume that we are given an entity $E = \langle N, \Gamma \rangle$ to be specified, where $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, $\alpha_i = \langle \Sigma_i, \sigma_i \rangle$. Assume also that the interaction points have separate alphabets: $\Sigma_i \cap \Sigma_j = \varnothing$ for $i \neq j$. We will denote the set $\Sigma_i^\tau = \Sigma_i \cup \{\tau\}$. An elementary structure will refer to the symbol transition $c \in \Sigma_1^\tau \times \Sigma_2^\tau \times \ldots \times \Sigma_k^\tau$, i.e., the transition is specified not by a single name but by a set of names $(a_1^\tau, a_2^\tau, \ldots, a_k^\tau)$, where $a_i^\tau \in \Sigma_i^\tau$. Since alphabets Σ_i do not intersect, the symbol transitions can be specified as sets of names, where the symbol τ can be omitted. Thus, transition t_1 in Fig. 3 is specified by set $\{T\_DAT, N\text{-}DAT\}$, while transition t_2 is specified by set $\{N\_DAT, L\_DAT\}$. The structure of the protocol entity is specified by means of the class of elementary structures and by the following operations; superposition (,); marking (n()); iteration (*); adjunction (;); elimination (□); and dissolution (○) [6].

Example 1. Disconnection procedure for X.25.3. Figure 4 shows the entity of the X.25.3 disconnection procedure, whose formal description may be found in [15]. The entity has four interaction points, N, L, P. TM, corresponding to services offered and used, to the protocol, and to timer operation. The points are defined on the corresponding alphabets $\Sigma_N = \{\downarrow N\_DIS, \uparrow N\_DIS\}$, $\Sigma_L = \{\downarrow L\_DAT, \uparrow L\_DAT\}$, $\Sigma_P = \{\downarrow CL\_R, \uparrow CL\_R, \downarrow CL\_C, \uparrow CL\_C\}$, $\Sigma_{TM} = \{ON, OFF, EXP\}$. Here it is assumed that protocol commands are transmitted by means of the data primitives of the link layer ↓L_DAT and ↑L_DAT. For example, execution of the "Request Disconnection" service primitive ↓N_DIS corresponds to execution of the protocol command ↓CL_R, which is transmitted in the primitive ↓L_DAT. The timer is activated in accordance with the procedure. Thus, activation of transition t_1 is interpreted as execution of four different actions at different interaction points. Activation of transition t_2 will correspond to execution of the protocol command ↑CL_C, which arrives with the primitive ↑L_DAT, and also deactivation of the timer. This transition is invisible for the service offered, i.e., for the higher-layer entity.

Example 2. Use of timer. Figure 5 shows the structure of a widely employed construct, known as a timer. A timer is an entity (TIMER) that has one interaction point T defined on alphabet $\Sigma_{TM} = \{ON, OFF, EXP\}$, where ON and OFF correspond to activation and deactivation of the timer respectively, while EXP corresponds to expiration of the time-out. In formula notation, the timer structure has the form $TIMER = (*(ON; EXP)) \circ OFF$. Assume that E is the entity of the disconnection procedure from the preceding example. Use of the timer by this entity is represented as the composition $(E_{TM} \| _{TM} TIMER)$.

Example 3. Transport service. Assume that we have formalized entities of the transport, network, link, and physical layers, denoted as $E^4(T, N)$, $E^3(N, L)$, $E^2(L, P)$, $E^1(P)$. Here T, N, L, P are interaction points corresponding to transport, network, link, and physical service primitives. Then the transport service can be represented as a single entity: $E_{TS} = E^4 \|_N E^3 \|_L E^2 \|_P E^1$. The transport-service entity has one interaction point, corresponding to the transport service. The remaining points become "invisible" after composition.

OPERATIONS WITH PARAMETERS

Up to now, only the control structure of entities has been considered. At the same time, operations with parameters and variables play an important part in composition of entities. Synchronization of entities may involve

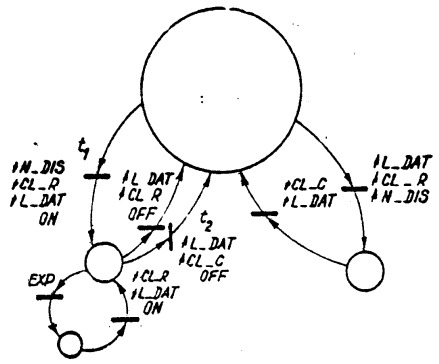

Fig. 4

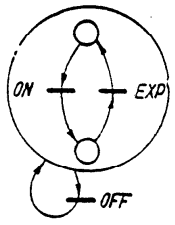

Fig. 5

Fig. 4. Entity of X.25/3 disconnection procedure.

Fig. 5. Timer structure.

exchange of data or harmonization of parameters. Paper [15] introduced, for the purpose of providing a complete protocol description, a formal model comprising a composition of high-level Petri nets (HLPN) and regular macronets. This model will be used in what follows to describe operations with parameters in protocol entities.

Assume that $\mathbf{A} = \{A_1, A_2, \ldots\}$ is a set of names of attributes, each of which A_i is specified on set $D(A_i)$. The HLPN comprises an interpretation in which a marker is associated with an attribute vector, while transition t_i is associated with the exciting predicate H_i and the procedure for calculating the attributes of the output markers G_i. An entity refers to the combination $E = \langle N', \Gamma' \rangle$, where N' is an HLPN, while each symbol $a_i \in \Sigma$ can have attribute vector $a_i \langle A_{i1}, A_{i2}, \ldots, A_{ik} \rangle$. Transition attributes A_{ij} can be attributes of markers of input and/or output places of the transition. Upon activation of a transition, attributes A_{ij} are assigned certain particular values. Thus, the activation of a particular transition differs in terms of the values of the attributes. Now, at an interaction point of the entity, the values of the corresponding parameters and variables are accessible in addition to certain transitions.

Attribute A_{ij} of transition t_i with name a_i will be called free if procedure G_i does not have rules for calculating attribute A_{ij}, i.e., it is assumed that A_{ij} can take any value from $D(A_{ij})$. Otherwise attribute A_{ij} will be called nonfree. A free attribute models reception of a value from without, this value subsequently proceeding via the output marker to the entity.

Let us consider an elementary operation of composition of networks N^1 and N^2, i.e., merger of transitions t_1 and t_2, marked by the same symbol $a\langle A_i \rangle$. The resultant transition $t_1 \cup t_2$ is excited and can be activated when transitions t_1 and t_2 are excited and can be activated, and with the same values of the transition attributes A_i. More rigorously, assume that the transition attribute are defined as $A_i := F_1(\mathbf{A})$ for transition t_1 and $A_i := F_2(\mathbf{A})$ for transition t_2. Then for transition $t_1 \cup t_2$ we have $H = H_1 \wedge H_2 \wedge F_1(A) = F_2(A)$.

This mechanism ensures several schemes of interaction via attribute A_i. Assume that attribute A_i is free for transition t_2, but unfree for t_1. Then it is evident that merging of transitions t_1 and t_2 describes transfer of the value of attribute A_i from network N^1 to network N^2. Merger of transitions with several attributes may correspond to transfer of data in multiple directions simultaneously. This mechanism of data transfer is analogous to the transfer of value in CCS [19]. If attribute A_i is unfree in transitions t_1 and t_2, then only processes with the same value of attribute A_i can interact via transition $t_1 \cup t_2$. Local (interface) connections between entities in the same system can be described by means of these mechanisms.

STRUCTURE OF PROTOCOL ENTITY

A protocol entity that is specified by means of several sets of primitives usually has a complex structure. Even the entity of the relatively simple class-0 transport protocol of the ISO [21] is fairly complicated and difficult to understand. In what follows, we propose a method of structuring protocol entities that are represented as the

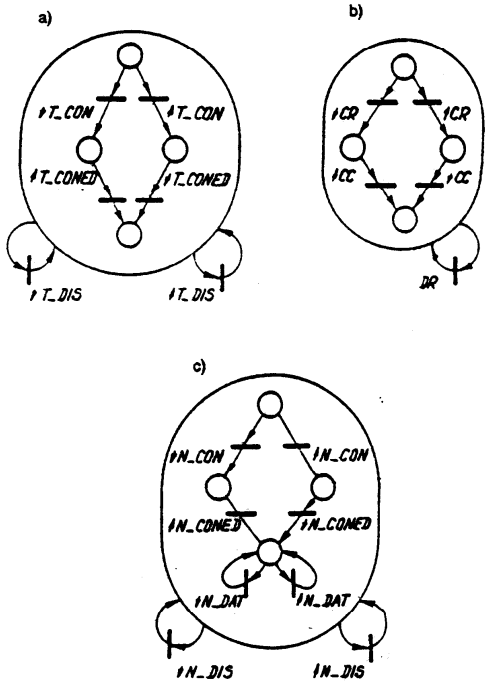

Fig. 6 Make-up of transport object entity: a) transport service; b) transport protocol; c) network service.

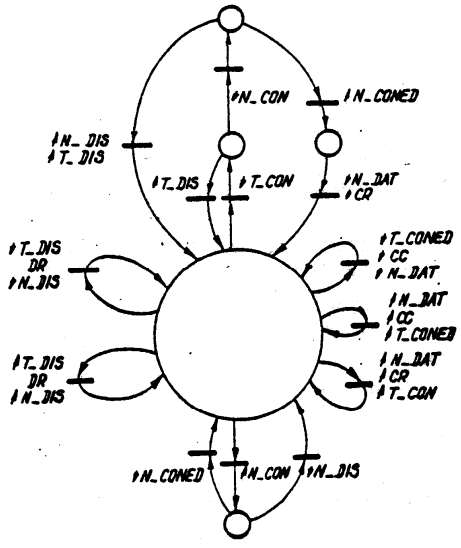

Fig. 7. Interface procedure of transport protocol.

composition of simpler entities. The method is illustrated using the example of the connection-establishment and disconnection procedures for the class-0 transport protocol of the ISO [21].

Assume that Σ_T={↓T_CON,↑T-CON,↓T_CONED,↑T_CONED,↓T_DIS,↑T_DIS}, Σ_N={↓N_CON,↑N_CON,↓N-CONED, ↑ N_CONED, ↓ N_DAT, ↑N_DAT, ↓N-DIS,↑N_DIS}, Σ_P={↓CR,↑CR,↓CC,↑CC,DR} are sets of names of primitives of the transport and network services, and of the transport protocol. Figure 6 shows the structures of the transport service, the transport protocol, and the network service. Figure 7 shows the entity of the interface procedure containing information on the relations between the execution of service and protocol commands. For example, after a request for establishment of a transport connection ↓T_CON, the entity initiates a network connection by the primitive ↓N_CON, and, after acknowledgment (↑N_CONED)it sends the protocol instruction↓CR by means of the primitive ↓N-DAT. If, however, there is a failure or refusal to establish a network connection (↑N_DIS), then there is a similar refusal to establish a transport connection (↑T_DIS). The protocol entity can be represented as the composition of these four entities. This representation readily yields a description in the form of a state graph, by setting up the marking graph for the corresponding Petri net. It is easy to establish that the marking graph for the entity in question is analogous to the transport-entity state graph that was given in [21].

CONCLUSIONS

We have introduced definitions of entities and of composition rules for them that enable us, first, to describe protocol entities; and, second, to combine protocol entities of different levels into a single unified hierarchical set. The concepts introduced will enable us subsequently to formulate and resolve problems of correctness of protocol sets, their implementation, and performance analysis. The establishment of connections between our model and protocol specification languages, as well as problems of automation of the process of development of protocol sets, will also be the subject of further studies.

REFERENCES

1. E. A. Yakubaitis, "The MAP/TOP/SAI set of network protocols," AVT [Automatic Control and Computer Sciences], no. 2, pp. 3-7, 1988.
2. R. Jardins, "Towards the information society: world cooperation on open systems standardization," in: Computer Network Usage: Recent Experience. Proc. IFIP TC 6 Working Conf. COMMET'85, pp. 15-17, 1986.
3. S. V. Rotanov and D. V. Kutsevalov, "Matching of network- and transport-layer services," in: Twelfth All-Union Seminar on Computer Networks [in Russian], part 2, pp. 9-13, Moscow-Odessa, 1987.
4. J. Harangozo, "Formal representation of the protocol hierarchy," Proc. Europ. Comput. Congr.: EUROCOMP'78, pp. 403-414.
5. M. Devy and M. Diaz, "Multilevel specification and validation of control in communication systems," First Int. Conf. on Distrib. Comput. Syst., pp. 43-50, 1979.
6. N. A. Anisimov, "Algebra of protocol structures based on the theory of Petri nets," AVT [Automatic Control and Computer Sciences], no. 1, pp. 9-15, 1987.
7. V. E. Kotov, "Algebra of regular Petri nets," Kibernetika, no. 5, pp. 10-18, 1980.
8. ISO 7498. Information Processing Systems. Open Systems Interconnection. Basic Reference Model, 1983.
9. G. Le Moli, S. Palazzo, and G. Andreoni, "A model of entity for the definition of protocols and interfaces," Protocol Specification. Testing and Verification, II: Proc. IFIP WG 6.1 Second Int. Workshop, pp. 249-258, 1982.
10. G. Janole, B. Algayers, and J. Dufau, "On communication protocol modeling and design," Lecture Notes in Comput. Sci., vol. 188, pp. 267-287, 1985.
11. W. Cellary, M. Saikowski, and M. Stroniski, "Defining a transport layer using numerical Petri nets," First Int. Conf. Comput. and Appl., pp. 353-360, Beijing, 1984.
12. H. J. Burhart, H. Eckert, and R. Prinoth, "Modeling of OSI communication services and protocols using predicate/transition nets," Protocol Specification. Testing and Verification, IV: Proc. IFIP WG 6.1 Fourth Int. Workshop, pp. 165-192, 1984.
13. S. S. Zaitsev et al., "Facilities for describing protocol architectures," in: Twelfth All-Union Seminar on Computer Networks [in Russian], part 2, pp. 20-24, Moscow-Odessa, 1987.

14. H. Koenig and M. Heiner, "The PDL system - a unified approach to the specification, verification, and implementation of protocols," Computer Network Usage: Recent Experience. Proc. IFIP TC 6 Working Conf. COMMET'85, pp. 15-17, 1986.

15. N. A. Anisimov, "Formal model for protocol development and description based on the theory of Petri nets," AVT [Automatic Control and Computer Sciences], no. 6, pp. 3-10, 1988.

16. N. A. Anisimov, "Formalization of protocol hierarchy on the basis of the theory of Petri nets," in: Packet-Switching Networks [in Russian], part 2, pp. 228-232, Riga, 1987.

17. G. V. Bochmann, "Finite state description of communication protocols," Computer Networks, vol. 2, pp. 4/5, pp. 361-372, 1978.

18. R. H. Cambell and A. N. Habermann, "The specification of processes synchronization by path expressions," Lecture Notes in Comput. Sci., vol. 16, pp. 89-102, 1974.

19. R. Milner, "A calculus for communication systems," Lecture Notes in Comput. Sci., vol. 92, 1980.

20. E.-R. Olderog and C. A. R. Hoare, "Specification-oriented semantics of communicating processes," Acta Informatica, vol. 23, pp. 9-66, 1986.

21. S. V. Rotanov and D. V. Kutsevalov, Class 0 Transport Protocol for Local-Area Networks [in Russian], parts 1-2, IEVT, Riga, 1985.

7 July 1988