A Synthesis Method for Fault-Tolerant Multicast Routing Protocol

Kenji Ishida and Kitsutaro Amano

Faculty of Information Sciences, Hiroshima City University,
3-4-1 Asaminami-ku, Hiroshima, 731-3166 Japan

Abstract. Multicast communication services will be one of the most promising future applications, which include real-time flows, in both the B-ISDN and the Internet. Design of such multicast routing protocols is complex and difficult due to complicated requirements. The protocol is defined to be fault-tolerant if messages can be retransmitted when a message loss occurs. In this paper, we propose a new synthesis method for generating a fault-tolerant multicast routing protocol for a given service specification, a network topology, and a tree shaped route.

1 Introduction

There are real-time tasks in which the computer system must respond to inputs within some maximum delay. If such a task is too large for the local computer, it might be handled by a number of remote computers working together. Such distributed computing will need the multicast communication service. Moreover, multicast communication services will be one of the most promising future applications, which include real-time flows, in both the B-ISDN and the Internet [4]. Generally, multicast communication can be realized by forwarding the messages along a tree shaped route. In this case, messages are duplicated by copy nodes on the tree. The protocol is defined to be fault-tolerant if messages can be retransmitted along a path when a message loss occurs. Fault-tolerance becomes an important characteristic to ensure quality of communication services [1], [2].

Design of practical multicast routing protocols is complex and difficult due to complicated requirements of fault-tolerance and synchronization of messages in the copy nodes. For such a difficult and complex protocol design, the protocol synthesis [5], [7] is recognized as one of the most prominent solutions, which automatically derives the protocol specifications without specification errors. In this paper, a synthesis of multicast routing protocols is defined as generation of a routing protocol specification from a routing service specification.

Up to the present, various protocol synthesis methods have been proposed [5], [6], [7], [8]. However, none of them was for multicast routing protocols in which behavior of copy nodes must be taken into consideration.

This paper proposes a new synthesis method for multicast routing protocols which satisfy the fault-tolerance. The proposed method generates a multicast routing protocol in consideration of behavior of a copy node. In the protocol,

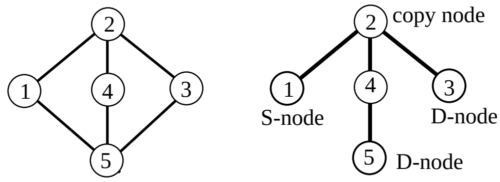

Fig. 1. A topology graph and treeshaped route

against a message loss, not a source node but a copy node can retransmit the message to destinations. Therefore, the retransmission can be fast.

The rest of this paper is organized as follows. Section 2 gives fundamental definitions concerning protocol synthesis. In Section 3, we define synthesis problem for multicast routing and proposed a solution of the problem. Then, we apply our method to a typical example. Finally, Section 4 concludes this paper.

2 System Model

2.1 Communication Model

A communication service is specified by service primitives exchanged between users in the higher layer and nodes in the lower layer through service access points (SAPs). A routing protocol can be viewed as a black box from users' view point. The nodes are also called protocol entities which are denoted by PEs in the following. Since users correspond to protocol entities in the higher layer, it is assumed that one user uses one PE. In a routing protocol, each PE must deliver a message through existing physical channels.

2.2 Topology Graph

Definition 1. A topology graph is defined as an undirected graph $G = (V, E)$, where V represents a set of PEs, and E $(\subseteq V \times V)$ represents a set of communication channels (FIFO queues).

For any two nodes PE_u, $PE_v \in V$ on a topology graph $G = (V, E)$, if there exists an edge $(u, v) \in E$, then node PE_u is called an adjacent node of PE_v.

Example 1. Figure 1 shows a topology graph and a treeshaped route from a *S-Node* to *D-nodes*. The number i $(1 \leq i \leq 5)$ in circle corresponds to PE_i $(1 \leq i \leq 5)$, respectively. In this paper, we consider the following multicast routing from a *S-Node* to *D-nodes*. First, PE_1 sends a message a to PE_2 (a copy node). Then, PE_2 receives a and sends $a1$ and $a2$ to PE_3 and PE_5, respectively, where $a1$ and $a2$ are copies of the message a. Then, PE_3 and PE_5 receive $a1$ and $a2$, respectively. Then, PE_3 and PE_5 send ACK or NACK to PE_2, respectively. Finally, PE_2 sends an ACK message to PE_1 only when it receives ACK messages from both PE_3 and PE_5, otherwise it sends a NACK message to PE_1.

2.3 Service Specification

In order to describe the behavior of a copy node efficiently, we consider two kinds of service specifications. One is a set of service specifications between a *S-node* and *copy node*s. Another is a set of service specifications between a *copy node* and *D-node*s. The synthesis method presented in this paper is applied to each of both specifications. As a result, several component pieces of protocol specifications are obtained. Finally, a final protocol specification is made of these pieces.

A service specification defines an execution order of service primitives which are exchanged between users and protocol entities through service access points. A service access point (SAP) between $user_i$ and PE_i is denoted by SAP_i.

Definition 2. A service specification is modeled by a Finite State Machine (FSM) $S =< S_s, \sum_s, T_s, \sigma >$ where

- S_s is a non-empty finite set of service states.
- $\sum_s$ is a finite set of service primitives. Each service primitive $p \in \sum_s \cup \{\varepsilon\}$ has, as an attribute, an index of service access point through which p passes, and ε is null primitive that causes no message exchanging. When p passes through SAP_i, we define a function $sap(p) = i$, and the primitive is denoted by p_i.
- T_s: $S_s \times \sum_s \rightarrow S_s$ is a partial transition function. For simplicity, we use T_s also to represent a set of triples (u, p, v) such that $v = T_s(u, p)$ $(u, v \in \sum_s)$.
- $\sigma \in S_s$ is an initial service state.

A state $u \in S_s$ is called a final state iff there is no outgoing transition (u, p, v) for any p and v. If more than one transition is outgoing from a service state, one of such transitions is chosen and executed. We call this FSM a service specification **S-SPEC**. An S-SPEC is represented by a labeled directed graph. For any state which represents a service state $s \in S_s$ in S, we define $OUT(S) = \{i|i = sap(p)\}$, where p is a label attached to an outgoing transition from s.

Example 2. An example of the S-SPEC is shown in Fig. 2 (a). In this figure, a circle denotes a service state, and an arrow denotes a transition between states. The state drawn by a bold circle is an initial state. This service specification represents sequences of message delivery from the source node to the destination node and its positive or negative acknowledgment from the destination node to the source node. For example, after $user_1$ sends S_req1 to PE_1 through SAP_1, $user_2$ receives S_ind2 from PE_2 through SAP_2 in this order. In case of $user_2$ sends S_call2 (ACK) through SAP_2, $user_1$ receives S_conf1 from PE_1 through SAP_1. In case of $user_2$ sends R_call2 (NACK) through SAP_2, $user_1$ receives S_conf1 from PE_1 through SAP_1.

2.4 Relation among Service Specifications

In order to handle the synchronization of messages in the copy node efficiently, we introduce **fork state** and **join state** into the protocol specification.

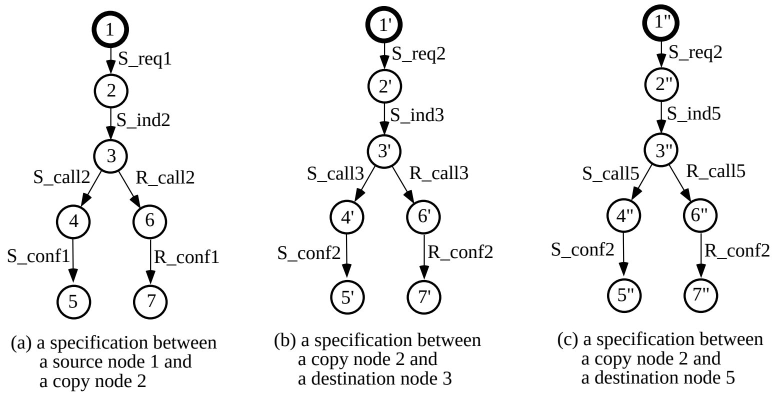

Fig. 2. Two kinds of service specifications (S-SPECs)

Definition 3. A state u is called a **fork state** iff there are several outgoing transition (u, p_1, v_1), (u, p_2, v_2) $\cdots$ (u, p_k, v_k) in the S-SPEC and the whole transitions are chosen and executed concurrently.

Definition 4. A state u is called a **join state** such that there are several incoming transition (v_1, p_1, u), (v_2, p_2, u), $\cdots$, (v_k, p_k, u) in the S-SPEC. And a state u is called an **or join state** such that u is a **join state** and an outgoing transition from u is executed when one of the incoming transition is executed. We assume that after the outgoing transition is executed, the **or join state** omits the other incoming transitions. And a state u is called an **and join state** such that u is a **join state** and an outgoing transition from u is executed only when the whole incoming transitions are executed.

Example 3. Figure 2 shows an example of the two kinds of specifications. In Fig. 2, (a) is a service specification between a *S-node* and a *copy node*, both (b) and (c) are service specifications between the *copy node* and *D-nodes*. A given relation among these specifications of Fig. 2 is depicted in Fig. 3.

2.5 Protocol Specification

The protocol specification consists of n-tuples of specifications for protocol entities. We assume that the adjacent nodes are determined by a given tree shaped route. Transmission and reception of messages between adjacent nodes are defined as follows.

Definition 5. If a message e is transmitted to PE_j, then it is denoted by a transmission event $!e(j)$. On the other hand, if a message e is received by PE_j, then it is denoted by a reception event $?e(j)$.

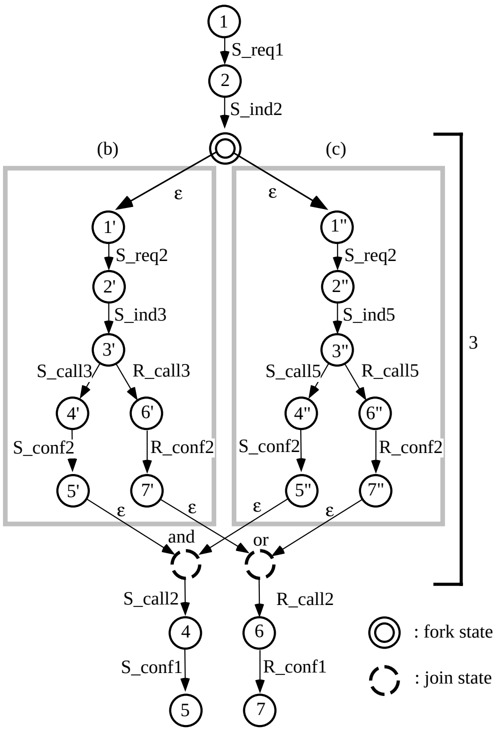

Fig. 3. Relation among S-SPEC (a), (b) and (c) in Fig.4

Definition 6. A protocol entity specification is modeled by an FSM $P_i = < S_{ip}$, $\sum_{ip}$, T_{ip}, $\sigma_{ip} >$ where

(1) S_{ip} is a non-empty finite set of protocol states.

(2) $\sum_{ip}$ is a non-empty finite set of protocol events. $\sum_{ip} = \{p | p \in \sum_s, sap(p) = i\} \cup MEX_i \cup \{$ T.O. $\} \cup \{\varepsilon\}$, where $\sum_s$ is a set of primitives in Definition 2, and MEX_i is a set of events which are transmitted to PE_{i1},PE_{i2}, $\cdots$, PE_{ik} or received by PE_{i1}, PE_{i2}, $\cdots$, PE_{il} and T.O is a timeout event that occurs when a predetermined time elapses. ε is null primitive that causes no message exchanging.

(3) $T_{ip} : S_{ip} \times \sum_{ip} \rightarrow S_{ip}$ is a partial transition function.

(4) $\sigma \in S_{ip}$ is an initial protocol state.

We call this FSM a PE-SPEC $_i$. As with the service specification, a protocol entity specification is also represented by a labeled directed graph. We explain a timeout transition(u, T.O, v) in T_{ip}. At the time when the state of PE-SPEC $_i$ moves to state u, counting time starts. Only when a current state of PE-SPEC $_i$ is state u and the predetermined time elapsed, the state of PE-SPEC $_i$ moves to state v. A transition "p/q" with p,$q \in T_p$ denotes a successive execution of transitions p and q.

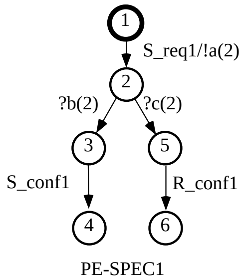

Fig. 4. Protocol entity specification PE-SPEC $_1$

Example 4. Figure 4 shows an example of PE-SPEC $_1$ for PE_1 in Fig.1. In this figure, a circle denotes a protocol state, and an arrow between states denotes a transition. For example, S_req1/!$a(2)$ implies that $user_1$ sends S_req1 to PE_1 through SAP_1, then PE_1 sends a message a to PE_2. The $?b(2)$ and $?c(2)$ imply that PE_1 receives a message b from PE_2 and PE_1 receives a message c from PE_2, respectively, where b corresponds to S_call2 (ACK) and c corresponds to R_call2 (NACK). In case of PE_1 receives b, a sequence $2 \rightarrow 3 \rightarrow 4$ is executed. In case of PE_1 receives c, a sequence $2 \rightarrow 5 \rightarrow 6$ is performed.

The following definition requires that messages are exchanged through existing channels on a tree T. If all transitions in T_{ip} obey channel restriction, we say P_i obeys channel restriction. And if all P_i obey channel restriction, we say P obeys channel restriction.

Definition 7. Consider a tree $T = (V', E')$ in a topology graph G and a protocol entity specification $P_i = < S_{ip}, \sum_{ip}, T_{ip}, \sigma_{ip} >$. Transitions $(u, !e(j), v)$ and $(u, ?e(j), v)$ in T_{ip} obey channel restriction if the following conditions are satisfied, respectively.

- If $(i, j) \notin E'$, then $(u, !e(j), v) \notin T_{ip}$ for any u, v, e, and
- If $(i, j) \notin E'$, then $(u, ?e(j), v) \notin T_{ip}$ for any u, v, e.

3 Synthesis Method for Multicast Routing

3.1 Protocol Synthesis Problem

During past few years, new applications emerged at LAN and Internet. The applications use multicast transmission, for resource discovery or multimedia conferences [4]. These applications require fault-tolerant multicast routing.

Protocol Synthesis Problem for fault-tolerant multicast routing to be solved in this paper is formally defined as follows:

Problem MR. Input A topology graph G, a tree shaped multicast route, and a service specification S. **Output** A multicast routing protocol specification P

Table 1. Transition synthesis rules

	Input	Condition	Output	
A1	$s1 \xrightarrow{Ei} s2$ PS-SPEC i	Out(s2)={ i }	$s1 \xrightarrow{Ei} s2$ PS-SPEC i	
B1	$s1 \xrightarrow{\varepsilon} s2$ PS-SPEC j ($j \neq i$)		$s1 \xrightarrow{\varepsilon} s2$ PS-SPEC j ($j \neq i$)	
A2	$s1 \xrightarrow{Ei} s2$ PS-SPEC i	Out(s2) $\neq$ { i }	$s1 \xrightarrow{Ei/!e(x)} s2$ x=Out(s2) PS-SPEC i	
B2	$s1 \xrightarrow{\varepsilon} s2$ PS-SPEC j ($j \neq i$)		$s1 \xrightarrow{?e(y)} s2$ y=Out(s1) PS-SPEC j ($j \in x$) x=Out(s2)	$s1 \xrightarrow{\varepsilon} s2$ PS-SPEC k ($k \notin x$) x=Out(s2)

which satisfies the following Conditions. **Condition 1** Unspecified receptions never occur in P. **Condition 2** Even if a message loss occurs, the execution order of service primitives defined by S is kept in P. **Condition 3** P obeys the channel restriction.

Nonexistence of unspecified receptions in Condition 1 and keeping execution order of service primitives in Condition 2 are ordinary conditions for protocol synthesis. Meanwhile the channel restriction based on a tree in Condition 3 and multicast routing with message retransmission are unique to our discussion.

3.2 Outline of Synthesis Method

For Problem MR, the proposed method to derive a protocol specification from a given service specification consists of the following three steps.

Step 1 Obtain projected service specification by applying the projection to the given each service specification S-SPEC. In these projected service specifications, the service primitives associated with SAP_i is represented by PS-SPEC $_i$ which is obtained from S-SPECs by substituting each transition not associated with SAP_i by ε.

Step 2 Construct PE-SPEC $_i$ by applying the transition synthesis rules shown in Table 1 to PS-SPEC $_i$ obtained at Step 1.

Step 3 Incorporate the capability of time out and retransmission into PE-SPEC $_i$ constructed at Step 2. Next, introduce the channel restriction to PE-SPEC $_i$. Next, remove ε transitions from PE-SPEC $_i$ by the algorithm in [3]. Finally, based on the resultant PE-SPEC $_i$ and the relation among the S-SPECs, we obtain the final protocol specification.

3.3 Example of Proposed Synthesis Method

In this subsection, we apply our synthesis method to a typical example. Consider a service specification S-SPECs shown in Fig. 2 and Fig. 3, and a topology graph G shown in Fig. 1.

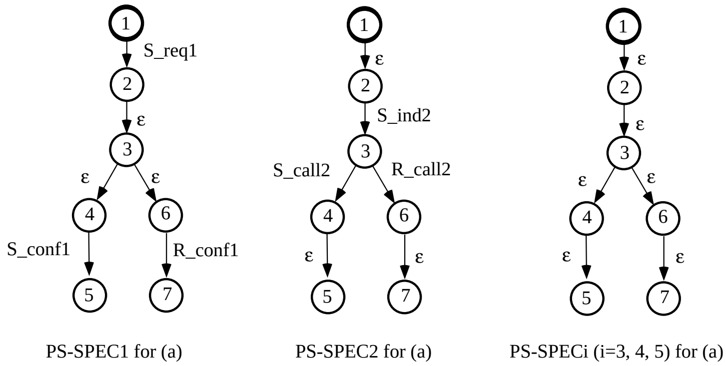

Fig. 5. Protocol specification PS-SPEC $_i$ for (a) after Step 1

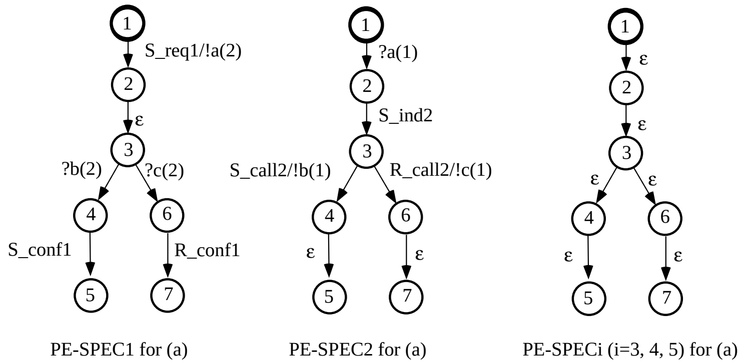

Fig. 6. Protocol specification PE-SPEC $_i$ for (a) after Step 2

At Step 1, service primitives are projected to PS-SPEC $_1$, PS-SPEC $_2$, PS-SPEC $_3$, PS-SPEC $_4$, PS-SPEC $_5$. For S-SPEC (a) in Fig. 2, the result of step 1 is shown in Fig. 5. Similarly, the other PS-SPEC $_i$ $(1 \leq i \leq 5)$ are obtained from S-SPEC (b) and (c).

At Step 2, based on the transition synthesis rules in Table 1, protocol entity specifications PE-SPEC $_i$ are obtained from PS-SPEC $_i$. For PS-SPEC $_i$ for (a) in Fig. 5, each PE-SPEC $_i$ for (a) in Fig. 6 shows the result of step 2. Similarly, the other PE-SPEC $_i$ are obtained from PS-SPEC (b) and (c).

At Step 3, firstly construct each PE-SPEC $_i$ that incorporates the capability of time out and retransmission into PE-SPEC $_i$ constructed at Step 2. For example, two time out transitions T.O. are added to PE-SPEC $_2$ in Fig. 7 and several self-loops are added to PE-SPEC $_i$ $(2 \leq i \leq 5)$ in Fig. 7. Next, introduce the channel restriction (**Condition 3**) to PE-SPEC $_i$. Intuitively, the channel restriction translates PE-SPEC $_i$ from an end-to-end communication into a link-to-link communication. For example, PE-SPEC $_4$ in Fig. 6 executes nothing. However,

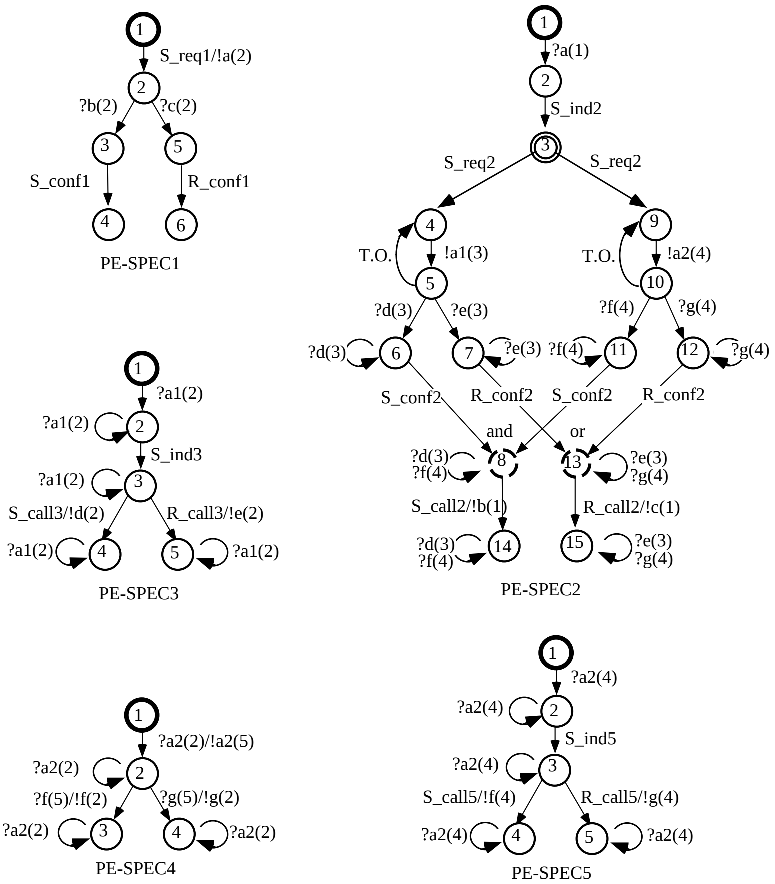

Fig. 7. Final protocol specification

PE-SPEC$_4$ in Fig. 7 relays messages between PE_2 and PE_5 because of the channel restriction (see Fig. 1). Next, remove ε transitions from each protocol entity specification by the algorithm [3]. Then, we obtain the PE-SPEC$_i$ ($1 \leq i \leq 5$) for S-SPEC (a), (b), (c). Finally, based on the relation among S-SPEC (a), (b), (c) in Fig. 3, we obtain a final protocol specification. Figure 7 shows the final protocol specification where each number in circle is renumbered.

3.4 Fault–tolerance of Routing Protocol

In this subsection, we discuss whether the protocol specification obtained by our method realizes fault–tolerance. Consider the protocol specification shown in Fig. 7, and assume that message loss between PE_2 and PE_3 in Fig. 1 occurs.

For example, PE_2 delivers a message a1 through the channel between PE_2 and PE_3. But this message is lost because of the transition failure of the link between PE_2 and PE_3. Then, PE_2 can know that the message is lost by the timeout event. Then, not the source node PE_1 but the copy node PE_2 retransmits the message (see the state 4 in PE-SPEC2 in Fig. 7). It is clear that for the transition link failure between PE_2 and PE_3, the execution order (**Condition 2**) of service primitives is kept. Moreover, the retransmission can be quickly.

For aspects of real-time system, messages must be delivered within the specified deadline. The allowable delay may vary from 10 msec for some real-time applications to 1 sec or less for interactive communication terminal. In this paper, a time out event T.O. is introduced. If a period of T.O. is too long or the number of message losses is too large, the deadline could not be guaranteed. Therefore, the determination of the period is an important issue for the protocol specification.

4 Conclusion

In this paper, we have proposed a new synthesis method which generates a fault-tolerant multicast routing protocol from a given service specification. The proposed method enables derivation of such a fault-tolerant protocol specification that messages are retransmitted at the copy node to the destination nodes even when a transition link failure occurs. Although we have assumed that at most one copy node on the tree shaped route, we can relax this assumption and we are currently trying to extend the proposed method.

References

1. Dolev S., Welch, J. L.: Crash resilient communication in dynamic networks. IEEE Trans. on Computers. **46** 1 (1997) 14–26
2. Herzberg, A.: Connection-based communication in dynamic networks. Proc. 11th Ann. ACM Symp. Principles of Distributed Computing (1992) 13–24
3. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Language, and Computation. Addison-Wesley (1979)
4. Huitema, C.: IPv6 The New Internet Protocol Second Edition. Prentice-Hall (1997)
5. Kakuda, Y., Nakamura, M., Kikuno, T.: Automated synthesis of protocol specifications with parallely executable multiple primitives. IEICE Trans. on Fundamentals. **E77-A** 10 (1994) 1634–1645
6. Liu, M. T.: Protocol Engineering. Advances in Computers **29** Academic (1989) 79–195
7. Probert R. L., Saleh K.: Synthesis of communication protocols: survey and assessment. IEEE Trans. on Computers. **40** 4 (1991) 468–476
8. Singh G., Sammeta M.: On the construction of multiphase communication protocols. Proc. International Conference on Network Protocols (ICNP94) (1994) 151–158