A Synthesis Method for Fault-tolerant and Flexible Multipath Routing Protocols

Yutaka Hatanaka, Masahide Nakamura, Yoshiaki Kakuda and Tohru Kikuno
Department of Informatics and Mathematical Science
Graduate School of Engineering Science, Osaka University
1-3, Machikaneyama-cho, Toyonaka-shi, Osaka 560, Japan
{hatanaka, masa-n, kakuda, kikuno}@ics.es.osaka-u.ac.jp

Abstract

Design of practical routing protocols is complex and difficult due to complicated requirements of fault-tolerance and flexibility. The protocol is defined to be fault-tolerant if messages can be rerouted via another path when the communication channel fails. In this paper, we propose a new synthesis method for generating a fault-tolerant routing protocol for a given service specification and a network topology. The routing protocol thus obtained adopts a multipath routing augmented with sets, where each set stores the next nodes for routing, and updates the sets according to network topology changes. Additionally, the routing protocol can attain flexibility by the multipath routing mechanism in the sense that only a small amount of changes is needed for addition or deletion of nodes. Finally, we show the effectiveness of the proposed method through an application to a typical routing service of message delivery from a source node to a destination node.

1 Introduction

Routing in a packet switched network is to deliver packets through communication paths from a source node to a destination node. It is desirable to route packets over the best possible communication paths available in the networks. The communication paths are determined by the network conditions such as queueing delay and processing time [10].

Routing that does not frequently change the communication paths is said to be static routing. Depending on the number of communication paths to be prepared between the source and destination nodes, static routing is divided into single path routing and multipath routing. The multipath routing is more robust than single path routing because as long as at least one of multiple paths between source and destination nodes is viable the messages will be delivered. Multipath routing is thus one of the most promising ways to realize the reliable routing services [3] [4].

The most fundamental requirement for multipath routing protocols is considered as follows:

Requirement (1) Fundamental capability for multipath routing: Since messages are delivered through multiple communication paths, protocol specification for the message delivery must be specified for any node on the communication paths.

Next, fault-tolerance and flexibility become important characteristics to ensure quality of communication services. Therefore, the protocol must also satisfy the following two hard requirements:

Requirement (2) Fault-tolerance for a communication channel failure: In order to surely deliver messages from source node to destination node even when a communication channel fails, the source node must possess a recovery function of rerouting.

Requirement (3) Flexibility for network topology changes: When some nodes and channels are newly added or deleted on the network, modification of the protocol specification must be easily done.

Since routing protocols become much larger and more complex due to the above hard requirements of fault-tolerance and flexibility, it is a serious problem to design routing protocols. For such a difficult and complex protocol design, the protocol synthesis [9] is recognized as one of the most prominent solutions, which

0-8186-8126-8/97 $10.00 © 1997 IEEE

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

automatically derives the protocol specifications without specification errors. In this paper, the synthesis of routing protocols is defined as generation of a routing protocol specification from a routing service specification, both of which are modeled by Finite State Machines (FSMs).

So far, various protocol synthesis methods have been proposed [12] [6] [8] [9]. However, none of them was for routing protocols with recovery function of rerouting, although the previous methods generate recovery functions such as retransmission for message loss, checkpointing and rollback recovery for coordination loss [1] [2] [11].

This paper proposes a new synthesis method for complex multipath routing protocols which satisfy the **Requirements (1), (2) and (3)**. The proposed method generates multipath routing augmented with sets, each of which is represented as t-set and stores the candidates of the next nodes. The sets are utilized for determining the next node to which messages are transmitted along a communication path. The synthesized protocol specification has a rerouting function, such that the messages can be rerouted through one of the multiple paths by referring the node in the t-set. Furthermore, the protocol specification can be modified easily just by updating nodes in the set, even when network topology changes.

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 multipath routing and proposed a solution of the problem. Then, we apply our method to a typical example in Section 4. Flexibility for topology change is discussed in Section 5. Finally, Section 6 concludes this paper with future research.

2 Preliminaries

2.1 Communication Model

As shown in Figure 1, 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**), and 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.

In a routing protocol, each PE must deliver a message through existing physical channels. Each channel between PEs is modeled by two unidirectional **FIFO** queues.

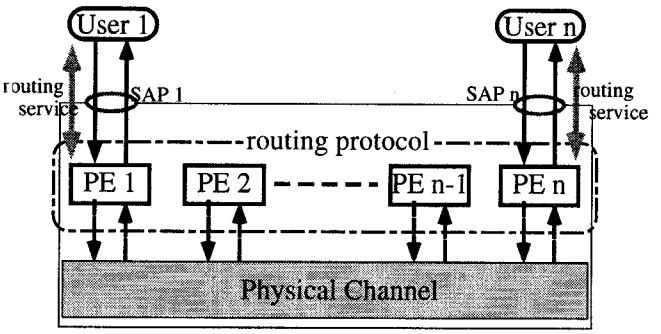

Figure 1: Communication model

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.

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.

This paper imposes the following restriction to assure the connectivity of the communication path between any pair of PEs even if a communication channel fails.

Restriction 1 : There are at least two edge-disjoint undirected paths between any two PEs in the topology graph.

From **Restriction 1**, for any pair of nodes $PE_i, PE_j \in V$, a path ρ between PE_i and PE_j must exist. Intuitively, the path ρ can be interpreted as a communication path from protocol entities PE_i to PE_j. Then, let us consider a case that $user_i$ communicates with $user_j$ via the path ρ. At first, $user_i$ sends the service primitive p to PE_i. Next, PE_i sends a message e to PE_j via the path ρ. Then, PE_j receives e and sends the service primitive q to $user_j$. For this, we call PE_i and PE_j, *S-node* and *D-node* of the path ρ, respectively. For the path ρ, the intermediate nodes between PE_i and PE_j are called *R-node*s on ρ. Messages are delivered from the S-node to the D-node via R-nodes on the communication path. As a special case, if PE_i is an adjacent node of PE_j and $\rho = (PE_i, PE_j)$, then ρ does not have R-node.

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

2.3 Service Specification

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 between $user_i$ and PE_i is denoted by $\mathbf{SAP}_i$.

Definition 2 : A service specification is modeled by a Finite State Machine (FSM) $\mathcal{S} =< S_{\mathcal{S}}, \Sigma_{\mathcal{S}}, T_{\mathcal{S}}, \sigma >$ where

- $S_{\mathcal{S}}$ is a non-empty finite set of service states.
- $\Sigma_{\mathcal{S}}$ is a finite set of service primitives. Each service primitive $p \in \Sigma_{\mathcal{S}}$ has, as an attribute, an index of service access point(**SAP**) through which p passes. When primitive p passes through SAP_i, we define a function $sap(p) = i$, and the primitive is denoted by p_i.
- $T_{\mathcal{S}} : S_{\mathcal{S}} \times \Sigma_{\mathcal{S}} \rightarrow S_{\mathcal{S}}$ is a partial transition function. For simplicity, we use $T_{\mathcal{S}}$ also to represent a set of triples (u,p,v)'s such that $v = T_{\mathcal{S}}(u,p)$ $(u,v \in S_{\mathcal{S}}, p \in \Sigma_{\mathcal{S}})$.
- $\sigma \in S_{\mathcal{S}}$ is an initial service state.

A state $u \in S_{\mathcal{S}}$ 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 an **S-SPEC**. A service specification **S-SPEC** is represented by a labeled directed graph. For each transition (u,p,v), we define vertices u and v for states u and v, respectively, and define an edge from vertex u to vertex v, and attach a label p to the edge. In the following, we refer to this edge as (u,p,v).

Consider an **S-SPEC** which represents a service specification $\mathcal{S} =< S_{\mathcal{S}}, \Sigma_{\mathcal{S}}, T_{\mathcal{S}}, \sigma >$. For any state which represents a service state $s \in S_{\mathcal{S}}$ in $\mathcal{S}$, we define $OUT(s) = \{i | i = sap(p)$ where p is a label attached to an outgoing transition from s $\}$.

An example of the **S-SPEC** is shown in Figure 2. In this figure, an oval denotes a service state, and an arrow denotes a transition between states. The state drawn by a bold oval 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 acknowledgement from the destination node to the source node. For example, after $user_1$ sends S_req1 to PE_1 through $\mathbf{SAP}_1$, $user_5$ receives S_ind5 from PE_5 through $\mathbf{SAP}_5$ in this order.

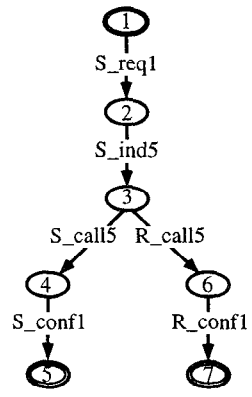

Figure 2: Service specification

Then, $user_5$ sends S_call5 through $\mathbf{SAP}_5$, and $user_1$ receives S_conf1 from PE_1 through $\mathbf{SAP}_1$.

This paper additionally imposes the following two restrictions to assure the correctness of the proposed protocol synthesis method.

Restriction 2 : The graph representation of **S-SPEC** is a tree.

This restriction implies that in the given **S-SPEC** at most one transition sequence exists between any two states. Based on the tree structure, we introduce the following execution order of service primitives.

Definition 3 : Consider an **S-SPEC** satisfying Restriction 2 and any transition sequence (u_1,p_1,u_2) $(u_2,p_2,u_3) \cdots (u_k,p_k,u_{k+1})$ in the **S-SPEC**, where u_1 is the initial state and u_{k+1} is a final state in **S-SPEC**. Then we define an execution order of service primitives $p_1, p_2, \cdots, p_k$, such that service primitive $p_i (i \leq k)$ must be executed before service primitives $p_{i+1}, p_{i+2}, \cdots, p_k$ for any i.

Restriction 3 : In the **S-SPEC**, for any three states u, v and w $(v \neq w)$, $T_{\mathcal{S}}$ does not include two transitions $(u,p,v),(u,p',w)$ with $sap(p) \neq sap(p')$ and $p \neq p'$.

This restriction implies that service primitives p and p' are not simultaneously exchanged through different **SAP**s.

2.4 Protocol Specification

Transmission and reception of messages between adjacent nodes are defined as follows.

Definition 4 : If message e is transmitted to PE_j, then it is denoted by a transmission event $!e(j)$. If message e is transmitted to one of PE_{j_1}, PE_{j_2}, $\cdots$,

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

PE_{j_k}, then it is denoted by a transmission event $!e(j_1,j_2,\cdots,j_k)$. On the other hand, if message e is received by PE_j, then it is denoted by a reception event $?e(j)$. If message e is received by one of PE_{j_1}, PE_{j_2}, $\cdots$, PE_{j_k}, then it is denoted by a reception event $?e(j_1,j_2,\cdots,j_k)$.

We introduce a set of nodes called *t-set* for each node PE_i. The *t-set* is used for determining the adjacent node to which the PE_i transmits or receives messages along a communication path. The transmission event $!e(t\text{-}set)$ where $t\text{-}set = \{j_1, j_2, \cdots j_k\}$ implies that message e is transmitted to one of the adjacent nodes PE_{j_1}, PE_{j_2}, $\cdots$, PE_{j_k}. We assume that the adjacent nodes are determined by the S-node of message m and the identification number of a communication path on which message m is delivered. This is so-called source-based routing. A reception event $?e(j_1, j_2, \cdots j_k)$ is also denoted by $?e(t\text{-}set)$ with $t\text{-}set = \{j_1, j_2, \cdots j_k\}$.

In order to realize loop-free transmission, we assume that after a message is received from adjacent PE_i, it cannot be transmitted to the same PE_i.

The protocol specification consists of n-tuples of specifications for protocol entities.

Definition 5 : A protocol entity specification is modeled by an FSM $P_i =< S_{ip}, \Sigma_{ip}, T_{ip}, \sigma_{ip} >$ where

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

(2) Σ_{ip} is a non-empty finite set of protocol events. $\Sigma_{ip} = \{p | p \in \Sigma_{\mathcal{S}}, sap(p) = i\} \cup MEX_i \cup \{T.O.\} \cup \{\varepsilon\}$, where $\Sigma_{\mathcal{S}}$ is a set of primitives in Definition 2, and MEX_i is a set of events which are transmitted from $PE_{i_1}, PE_{i_2}, \cdots, PE_{i_k}$ or received by $PE_{i_1}, PE_{i_2}, \cdots, PE_{i_l}$, 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}: $S_{ip} \times \Sigma_{ip} \rightarrow S_{ip}$ is a partial transition function. For simplicity, we also use T_{ip} to represent a set of triples (u,p,v) such that $v = T_{ip}(u,p)$.

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

We call this FSM a **PE-SPEC***i*. As for 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 state $u \in S_{ip}$ is called a *final state* iff there is no outgoing transition (u,p,v) for any p and v. A state $u \in S_{ip}$ is called a *receiving state* for e iff any outgoing transition form u is a reception event $(u, ?e(t\text{-}set), v)$ for any *t-set* and v.

Protocol specification (**P-SPEC**) P consists of n-tuples of $P_i =< S_{ip}, \Sigma_{ip}, T_{ip}, \sigma_{ip} >$ $(1 \leq i \leq n)$, and P is denoted by an FSM $P =< S_p, \Sigma_p, T_p, \sigma_p >$ where $S_p = (S_{1p} \times S_{2p} \times \cdots \times S_{np})$, $\Sigma_p = (\Sigma_{1p} \cup \Sigma_{2p} \cup \cdots \cup \Sigma_{np})$, $T_p = (T_{1p}, T_{2p}, \cdots, T_{np})$ and $\sigma_p = (\sigma_{1p}, \sigma_{2p}, \cdots, \sigma_{np})$.

Then, a transition "p/q" with $p, q \in T_p$ denotes a successive execution of transitions p and q.

An example of a protocol entity specification is shown in Figure 3. In this figure, an oval denotes a protocol state, and an arrow between states denotes a transition. For example, $!a(t\text{-}set)$ where $t\text{-}set = \{5,2\}$ implies two possible message transmissions $!a(5)$ and $!a(2)$.

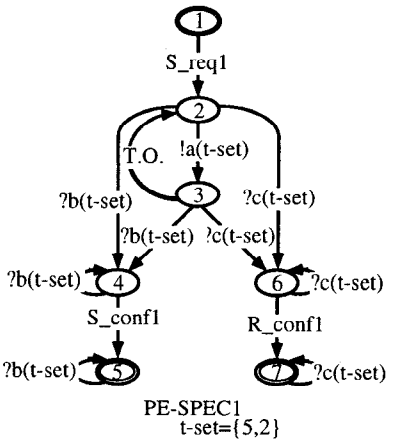

Figure 3: Protocol entity specification

In this paper, we focus on the following protocol errors called *unspecified receptions*. Although the same results are obtained for other protocol errors, discussions are omitted due to the limited page.

Definition 6 : Assume that message e sent by PE_i is on the top of an **FIFO** queue (that is, a communication channel) from PE_i to $PE_j (i \neq j)$. If a current state u of PE_j is a final state or is a receiving state for any $e' (\neq e)$, then we say that an *unspecified reception* with respect to e occurs in **PE-SPEC***j*.

The following definition requires that messages are exchanged through existing channels.

Definition 7 : Consider a topology graph $G = (V, E)$ and a protocol entity specification $P_i =< S_{ip}, \Sigma_{ip}, T_{ip}, \sigma_{ip} >$. Transitions $(u, !e(j), v)$ and

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

$(u, ?e(j), v)$ 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.

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.

3 Synthesis Method for Fault-tolerant Multipath Routing

3.1 Protocol Synthesis Problem

Protocol Synthesis Problem for Fault-tolerance to be solved in this paper is formally defined as follows:

Input: A topology graph G with Restriction 1, and a service specification $\mathcal{S}$ with Restrictions 2 and 3.

Output: A protocol specification P which satisfies the following Conditions 1, 2 and 3.

Condition 1 : Unspecified receptions never occur in P.

Condition 2 : Even if a communication channel fails, the execution order of service primitives defined by $\mathcal{S}$ is kept in P.

Condition 3 : P obeys the channel restriction.

Non existence of unspecified receptions in Condition 1 and keeping execution order of service primitives in Condition 2 are ordinary conditions for protocol synthesis. On the other hand, the channel restriction in Condition 3 and discussions on the failure of communication channel in Condition 2 are unique to our discussion. **Requirements (1) and (2)** in **Introduction** are taken into account as Conditions (1) through (3) in the cases with no communication channel failures and with communication channel failures, respectively.

3.2 Outline of Synthesis Method

The proposed method to derive a protocol specification from a given service specification consists of the following four steps.

Step 1 Obtain n projected service specifications by applying the projection to the given service specification.

Step 2 Construct an n protocol entity specifications by applying the transition synthesis rules shown in Table 1 (to be explained in **3.3.2**).

Step 3 Incorporate the capability of multipath routing into the protocol entity specifications constructed at Step 2, such that the resultant specification obeys channel restriction. Then remove ε transitions from each protocol entity specification by the algorithm in [5].

Step 4 Incorporate the recovery function of rerouting into the protocol specification constructed at Step3, using timeout event.

3.3 Detail of Synthesis Method

Since Step 1 and Step 2 are almost the same as those in [6], they will be explained briefly in the following.

3.3.1 Step 1

Projection is used for dividing a service specification into a set of projected service specifications. Each projected service specification corresponds to each PE. As in the case of **S-SPEC**, we use a labeled directed graph to represent each projected service specification.

In the projected service specification, the service primitive associated with **SAP***i* is represented only by **PS-SPEC***i*. Any service primitive that does not contribute to PE_i is substituted by ε.

In this step, each projected service specification **PS-SPEC***i*$(1 \leq i \leq n)$ is obtained from a service specification **S-SPEC** by substituting each transition not associated with **SAP***i* by ε.

3.3.2 Step 2

In this step, n protocol entity specifications **PE-SPEC**s are obtained from n projected service specifications **PS-SPEC**s. This transformation is performed by applying the transition synthesis rules shown in Table 1.

In Table1, OUT is specified in **2.3**, and E_i denotes a service primitive in the **PS-SPEC***i*. Each pair of transition synthesis rules Ak and $Bk(k = 1, 2)$ is applied to each pair of transitions (S_1, E_i, S_2) in **PS-SPEC***i* and (S_1, ε, S_2) in **PS-SPEC***j*$(j \neq i)$, respectively. Message e is uniquely generated for each service primitive E_i in the rules $A2$ and $B2$.

In **PE-SPEC**s obtained at Step 2, transitions for messages to be transmitted and to be received directly between the S-node and the D-node are generated. If there is a path from the S-node to D-node having no

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

R-nodes, these transitions obey the channel restriction. However, in general, the result violates the channel restriction.

Table 1: Transition synthesis rules

	Input	Condition	Output
A1	S1 —Ei→ S2 PS-SPECi	Out(S2)={i}	S1 —Ei→ S2 PE-SPECi
B1	S1 —ε→ S2 PS-SPECj(≠ i)		S1 —ε→ S2 PE-SPECj(≠ i)
A2	S1 —Ei→ S2 PS-SPECi	Out(S2)≠{i}	S1 —Ei/!e(X)→ S2 PE-SPECi X=Out(S2)
B2	S1 —ε→ S2 PS-SPECj(≠ i)		S1 —?e(X)→ S2 PE-SPECj (j∈X); S1 —ε→ S2 PE-SPECk (k∉X)

3.3.3 Step 3

Step 3 incorporates the capability of multipath routing into n protocol entity specifications **PE-SPEC**s that obey the channel restriction. This incorporation is executed by applying the following **TE procedure**.

Note that the protocol specification obtained at Step 2 possesses the same graph structure as the service specification, because the transition synthesis rule adds neither states nor transitions, and removes neither states nor transitions. That is, each **PE-SPEC***i* has the same number of states and transitions. So, for a transition (u, E_i, v) in **S-SPEC**, there exist n transitions such that $(u, E_i/!e(j), v) \in T_{ip}$, $(u, ?e(i), v) \in T_{jp}$, and n-2 transitions $(u, \varepsilon, v) \in T_{kp}$ $(k \neq i, j)$.

TE procedure is applied to such n transitions.

TE procedure:

Input: **PE-SPEC**s obtained at Step2, and topology graph $G = (V, E)$.

Output: **PE-SPEC**s with the capability of multipath routing that obeys channel restriction.

Procedure: For each n transitions $(u, E_i/!e(j), v) \in T_{ip}$, $(u, ?e(i), v) \in T_{jp}$, and $(u, \varepsilon, v) \in T_{kp}$ $(k \neq i, j, 1 \leq k \leq n)$, execute TE-Step 1 through TE-Step 4. Then remove all ε transitions using the algorithm in [5].

TE-Step 1 Search all loop-free paths $\rho_1, \cdots \rho_m$ from PE_i to PE_j based on G.

TE-Step 2 If $(i, j) \notin E$, remove transitions $(u, E_i/!e(j), v)$ and $(u, ?e(i), v)$ from T_{ip} and T_{jp}, respectively.

TE-Step 3 For each $\rho \in \{\rho_1, \cdots, \rho_m\} - \{\rho_d\}$, where ρ_d is a path from PE_i to PE_j having no R-nodes, execute TE-Substeps 3.1 and 3.2.

TE-Substep 3.1 For each T_{kp} $(k \neq i, j)$ such that PE_k is a R-node on ρ, remove (u, ε, v) from T_{kp}.

TE-Substep 3.2 Add several transitions for each PE based on ρ as follows:

(1) For an adjacent node PE_x of PE_i on ρ (PE_i is an S-node on ρ), add a transition $(u, E_i/?e(x), v)$ to T_{ip}.

(2) For an adjacent node PE_y of PE_j on ρ (PE_j is a D-node on ρ), add a transition $(u, ?e(y), v)$ to T_{jp}.

(3) For each T_{kp} $(k \neq i, j)$ such that PE_k is a R-node on ρ and PE_z and PE_w are a pair of adjacent nodes of PE_k, add a transition $(u, ?e(z)/!e(w), u)$ to T_{kp} where PE_z is on the sub-path of ρ from PE_i to PE_k, and PE_w is on the sub-path of ρ from PE_k to PE_j.

TE-Step 4 Introduce *t-set* into **PE-SPEC** as follows:

(1) In the T_{ip}, remove transitions $(u, E_i/!e(x_1), v)$, $\cdots$, $(u, E_i/!e(x_m), v)$. Next, create a new state u' in S_{ip}, then add two transitions (u, E_i, u') and $(u', !e(t\text{-}set), v)$. Finally, let $t\text{-}set = \{x_1, \cdots, x_m\}$.

(2) In the T_{jp}, remove transitions $(u, ?e(y_1), v)$, $\cdots$, $(u, ?e(y_m), v)$ and add transition $(u, ?e(t\text{-}set), v)$. Then let $t\text{-}set = \{y_1, \cdots, y_m\}$.

(3) In the T_{kp} such that PE_k is a R-node on a path $\rho \in \{\rho_1, \cdots, \rho_m\}$, remove all transitions $(u, ?e(z_1)/!e(w_1), u)$, $\cdots$, $(u, ?e(z_{m'})/!e(w_{m'}), u)$ and add transitions $(u, ?e(t\text{-}set1)/!e(t\text{-}set2), u)$. Then let $t\text{-}set1 = \{z_1, \cdots, z_{m'}\}$ and $t\text{-}set2 = \{w_1, \cdots, w_{m'}\}$. Here, $1 \leq m' \leq m$.

At TE-Step 1, loop-free communication paths from the S-node to the D-node for the message are searched as much as possible by the conventional path enumeration method [7]. At TE-Step 2, transitions that violate channel restriction are removed. If ρ_d exists in G, it is clear that the transitions $(u, E_i/!e(j), v) \in T_{ip}$ and $(u, ?e(i), v) \in T_{jp}$ obey channel restriction. It is not necessary to execute TE-Step 2 and TE-Substeps 3.1 and 3.2 for ρ_d. In the TE-Step 4, modification of the S-node is done to avoid transmission of E_i more than once.

3.3.4 Step 4

In this step, we incorporate the function of rerouting into protocol entity specifications **PE-SPEC**s obtained at Step 3. When a communication channel fails,

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

a protocol entity at the source node finds the failure by the timeout event, and retransmits messages.

When a channel failure occurs on a path and a transmitted message is lost, reception events are not executable in PEs on the path. This is because the source node forever waits for receiving acknowledgement of the transmitted message. So, we add transitions for retransmission of the message to the source node. However, as a side effect, unspecified receptions may occur. Therefore, we add supplementary transitions so that unspecified receptions do not occur.

The procedure of Step4 consists of the following steps S1, S2 and S3: For each transition sequence from the initial state u_{init} to the final state in each **PE-SPEC**, we apply steps S1-S3 in this order.

(S1) Search a transition (u_0, E, u_1) such that E is either a transmission event or a reception event and all transitions from u_{init} to u_0 are service primitives.

(S2) Consider two cases: $E = !e_1(x_1)$ and $E = ?e_1(x_1)$.

(Case of $E = !e_1(x_1)$)

Assume that the event $(u_n, ?e_2(x_2), u_{n+1})$ is the first reception event on the transition sequence from u_0. Then, we add the following transitions (1), (2) and (3) to **PE-SPEC**i.

(1) $(u_n, T.O., u_0)$.

(2) $(u_0, ?e_2(x_2), u_{n+1})$, $(u_1, ?e_2(x_2), u_{n+1})$ $\cdots$, $(u_{n-1}, ?e_2(x_2), u_{n+1})$.

(3) $(v, ?e_2(x_2), v)$ for each state v included in the transition sequence from u_n to the final state u_m.

After this, we regard u_{n+1} as u_{init}.

(Case of $E = ?e_1(x_1)$)

Assume the event $(u_k, ?e_3(x_3), u_{k+1})$ is the first reception event on the execution sequence from u_1. Then, we perform the followings.

Let the event $(u_n, !e_2(x_2), u_{n+1})$ be the first transmission event on the transition sequence from state u_1 to u_k. (If the transition is not found, we regard u_n as u_k.) Then, we add the following transitions (1) and (2) to **PE-SPEC**i. (If $u_n = u_k$, we only add a transition (1).)

(1) $(u_1, ?e_1(x_1), u_1)$, $(u_2, ?e_1(x_1), u_2)$, $\cdots$, $(u_n, ?e_1(x_1), u_n)$.

(2) $(u_{n+1}, ?e_1(x_1), u_n)$, $(u_{n+2}, ?e_1(x_1), u_n)$, $\cdots$, $(u_k, ?e_1(x_1), u_n)$.

After this, we regard u_k as u_{init}.

(S3) We recursively execute (S1) and (S2) from the new u_{init}.

4 Example

4.1 Synthesis of Multipath Routing Protocol

In this section, we apply our synthesis method to a typical example. Consider a service specification **S-SPEC** shown in Figure 2 and a topology graph **G** shown in Figure 4.

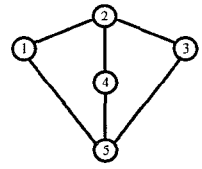

Figure 4: Topology graph

At Step1, service primitives are projected to **PS-SPEC1**, **PS-SPEC2**, **PS-SPEC3**, **PS-SPEC4**, **PS-SPEC5**.

At Step2, protocol entity specifications **PE-SPEC**s are obtained from **PS-SPEC**s. For example, consider transition $(1, S\_req1, 2)$ in **PS-SPEC1** and transition $(1, \varepsilon, 2)$ in **PS-SPEC**i $(i = 2,3,4,5)$. For this case, since $OUT(2) = \{5\} \neq \{1\}$, the transition synthesis rules $A2$ and $B2$ are applied. As a result, two transitions $(1, S\_req1, 2)$ in **PS-SPEC1** and $(1, \varepsilon, 2)$ in **PS-SPEC5** are changed to transitions $(1, S\_req1/!a(5), 2)$ and $(1, ?a(1), 2)$ respectively. But $(1, \varepsilon, 2)$ in **PS-SPEC**i $(i = 2,3,4)$ remains unchanged since $\{2, 3, 4\} \neq OUT(s)$ for any $s \in S_s$. Figure 5 shows the result of Step2.

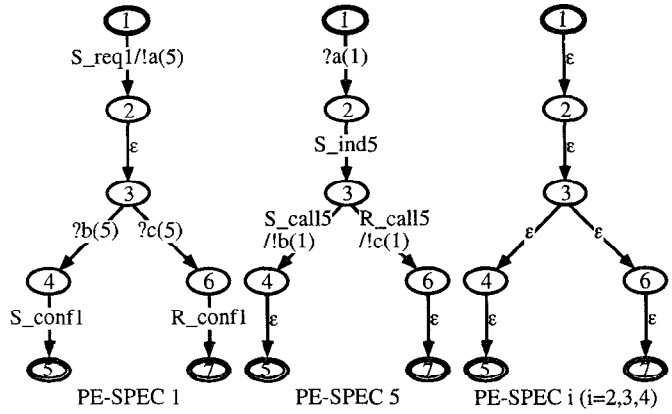

Figure 5: Protocol specification after Step 2

Step3 constructs **PE-SPEC**s with multipath transmission that obey the channel restriction. For example, consider transitions $(1, S\_req1/!a(5), 2)$ in **PE-SPEC1**, $(1, ?a(1), 2)$ in **PE-SPEC5**, $(1, \varepsilon, 2)$ in **PE-SPEC**Si $(i = 2,3,4)$ in Figure 5. There are three

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

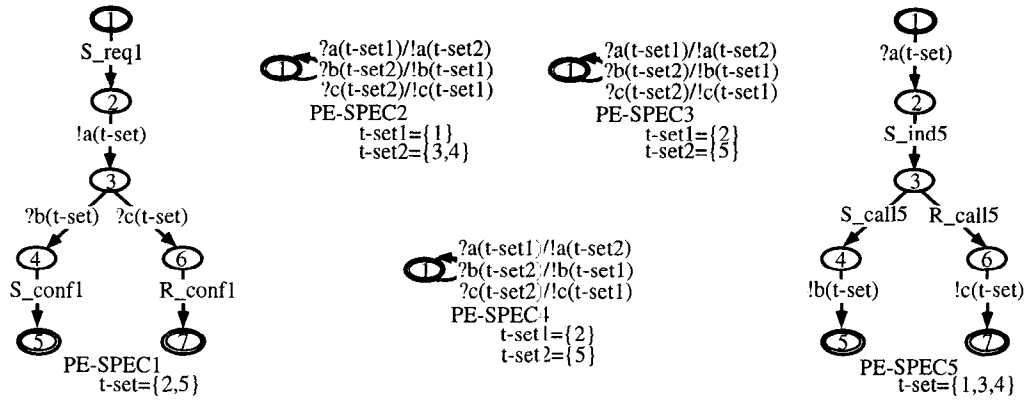

Figure 6: Protocol specification after Step3

paths from node1 (PE_1) to node5 (PE_5) in $\mathbf{G}$: ρ_1: $PE_1 \rightarrow PE_5$, ρ_2: $PE_1 \rightarrow PE_2 \rightarrow PE_3 \rightarrow PE_5$ and ρ_3 : $PE_1 \rightarrow PE_2 \rightarrow PE_4 \rightarrow PE_5$. Next, with respect to ρ_1, transitions $(1, S\_req1/!a(5), 2)$, $(1, ?a(1), 2)$ are unchanged in **PE-SPEC**i $(i = 1, 5)$, respectively since ρ_1 has no R-node. For ρ_2, transitions $(1, S\_req1/!a(2), 2)$, $(1, ?a(1)/!a(3), 1)$, $(1, ?a(2)/!a(5), 1)$, $(1, ?a(3), 2)$ are added to **PE-SPEC**i $(i = 1, 2, 3, 5)$, By TE-Substep 3.2, the transitions in **PE-SPEC**1 are modified to transitions $(1, S\_req1, 1')$, respectively. Similarly for ρ_3, transitions $(1, S\_req1/!a(2), 2)$, $(1, ?a(1)/!a(4), 1)$, $(1, ?a(2)/!a(5), 1)$, $(1, ?a(4), 2)$ are added to **PE-SPEC**i $(i = 1, 2, 4, 5)$, respectively. $(1', !a(t\text{-}set), 2)$ where $t\text{-}set = \{5, 2\}$. Similarly, the transitions in **PE-SPEC**5 are modified to $(1, ?a(t\text{-}set), 2)$ where $t\text{-}set = \{1, 3, 4\}$, and the transitions in **PE-SPEC**2 are also modified to $(1, ?a(t\text{-}set1)/!a(t\text{-}set2), 1)$ where $t\text{-}set1 = \{1\}$ and $t\text{-}set2 = \{3, 4\}$. Other transitions in **PE-SPEC**3 and **PE-SPEC**4 are similarly modified, and all ε transitions are removed. Figure 6 shows a protocol specification **PE-SPEC**s after Step3.

At Step4, timeout events and some other transitions are added. For example, consider execution sequence $S\_req1, !a(t\text{-}set), ?b(t\text{-}set), S\_conf1$ in **PE-SPEC**1. Since transition $(3, ?b(t\text{-}set), 4)$ exists after transition $(2, !a(t\text{-}set), 3)$ on the transition sequence, four transitions $(3, T.O., 2)$, $(2, ?b(t\text{-}set), 4)$, $(4, ?b(t\text{-}set), 4)$, $(5, ?b(t\text{-}set), 5)$ are added in **PE-SPEC**1. Figure 7 shows the resultant protocol specification after Step4.

4.2 Fault-tolerance

In this subsection, we discuss whether the protocol specification obtained by our method realizes both **Requirements (1) and (2)** or not.

Consider the protocol specification shown in Figure 7, and assume that the channel between PE_1 and PE_5 fails.

First PE_1 delivers directly message **a** through the channel between PE_1 and PE_5. But this message is lost because of the channel failure. Then, PE_1 can know that the message is lost by the timeout event.

Next, the PE_1 retransmits the message via another path. Let us consider the following path : $PE_1 \rightarrow PE_2 \rightarrow PE_3 \rightarrow PE_5$. This retransmission is realized by executing transitions $!a(t\text{-}set)$ in **PE-SPEC**1, $?a(t\text{-}set1)/!a(t\text{-}set2)$ in **PE-SPEC**2, $?a(t\text{-}set1)/!a(t\text{-}set2)$ in **PE-SPEC**3 and $?a(t\text{-}set)$ in **PE-SPEC**5. Here, $t\text{-}set$ and $t\text{-}set2$ are interpreted as follows. Based on the source-based routing policy in Definition 4, for example, $!a(2)$ is actually executed for $!a(t\text{-}set)$ in **PE-SPEC**1, and $!a(3)$ is executed for $!a(t\text{-}set2)$ in **PE-SPEC**2.

It is clear that for the channel failure between PE_1 and PE_5, the execution order of service primitives is kept in the transition sequence denoted by bold arrows in Figure 7.

5 Flexibility for Topology Change

5.1 Problem

Corresponding to **Requirement (3)**, we define a new kind of updating problem as follows:

Input: (1) A protocol specification which is obtained by applying our synthesis method to a topology graph G.

(2) Topology change ΔG due to addition or deletion of a node and its associated channels (We assume that the updated graph $G + \Delta G$ or $G - \Delta G$ still satisfies Restriction 1).

Output: A protocol specification, which is obtained by applying our synthesis method to the topology graph $G + \Delta G$ or $G - \Delta G$.

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

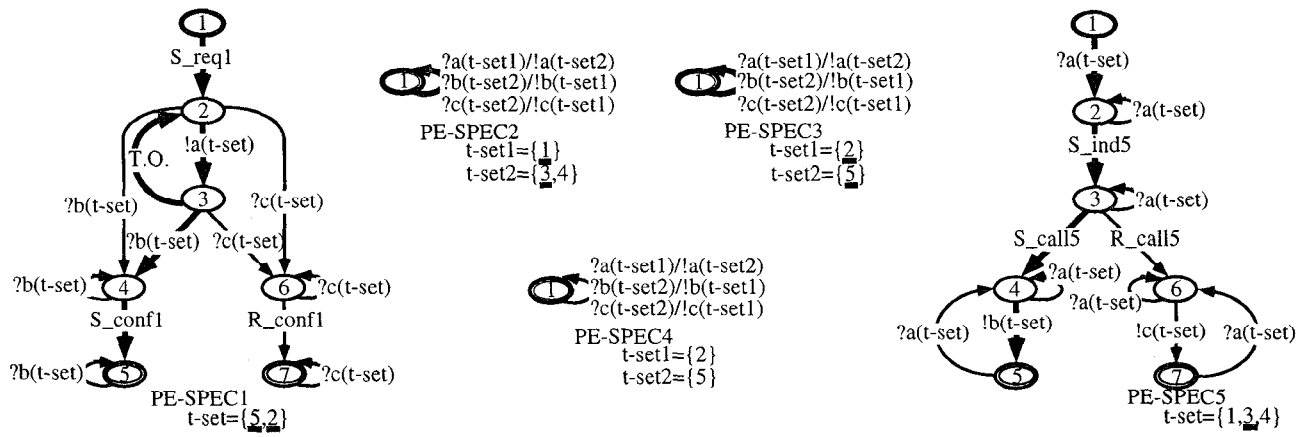

Figure 7: Final protocol specification

In the next section, we propose the following modification method which obtains the output effectively using *t-set* rather than applying directly our synthesis method to $G + \Delta G$ or $G - \Delta G$.

5.2 Modification Method

In this subsection, we describe only a procedure that realizes **Requirement (3)**. At first, we explain the case of $G + \Delta G$ (that is some PEs are added).

We suppose that PEs and their associated channels are incrementally added.

Assume that PE_i is a newly added node and $PE_{j_1}, PE_{j_2}, \cdots, PE_{j_k}$ are connected to PE_i. These are the elements of ΔG. The added PE_i can be R-node. Then, let PE_{j_p} and PE_{j_q} $(1 \leq p \leq k, 1 \leq q \leq k, p \neq q)$ are arbitrary two PEs among $PE_{j_1}, PE_{j_2}, \cdots, PE_{j_k}$ as shown in Figure 8. There exist loop-free directed paths from S-node to D-node through PE_{j_p} and PE_{j_q} in G. For the PEs on these paths, transitions for message delivery have been specified in **PE-SPEC**s.

By adding PE_i and channels (i, j_p), (i, j_q), new loop-free directed paths, each of which consists of a subpath from S-node to PE_{j_p}, a subpath (j_p, i), (i, j_q), and a subpath from PE_{j_q} to D-node, appear in $G + \Delta G$. Since, for the PEs on the subpath from S-node to PE_{j_p} and the subpath from PE_{j_q} to D-node, transitions for message delivery have been specified in **PE-SPEC**s, modification of **PE-SPEC** is unnecessary for the PEs except for PE_{j_p} and PE_{j_q}.

We have only to change, for each message e, !e(t-set_p) in **PE-SPEC**j_p and ?e(t-set_q) in **PE-SPEC**j_q into !e(t-set'_p) where $t\text{-}set'_p = t\text{-}set_p \cup \{i\}$ and ?e(t-set'_q) where $t\text{-}set'_q = t\text{-}set_q \cup \{i\}$, respectively. Similarly, if transition ?e(t-set_{i1})/!e(t-set_{i2}) does not exist in **PE-SPEC**i, then transition ?e(t-set_{i1})/!e(t-set_{i2}) where $t\text{-}set_{i1} = \{j_p\}$ and $t\text{-}set_{i2} = \{j_q\}$ is added. Otherwise the transition is updated to ?e(t-$set_{i1} \cup \{j_p\}$)/!e(t-$set_{i2} \cup \{j_q\}$).

Next, we explain the case of $G - \Delta G$. Since deletion of S-node or D-node makes message delivery impossible, we assume that the deleted PE is R-node. Assume that in Figure 8 PE_i is a deleted node and that PE_i was connected to $PE_{j_1}, PE_{j_2}, \cdots, PE_{j_k}$. By eliminating PE_i and channels (i, j_p), (i, j_q), directed paths (j_p, i), (i, j_q) are deleted for any p and q $(1 \leq p \leq k, 1 \leq q \leq k, p \neq q)$.

In this case, we have only to delete from **PE-SPEC**j_p index $i \in t\text{-}set_p$ in !e(t-set_p) and we also delete from **PE-SPEC**j_q index $i \in t\text{-}set_q$ in ?e(t-set_q). Then, we delete ?e(t-set_{i1})/!e(t-set_{i2}) in **PE-SPEC**i.

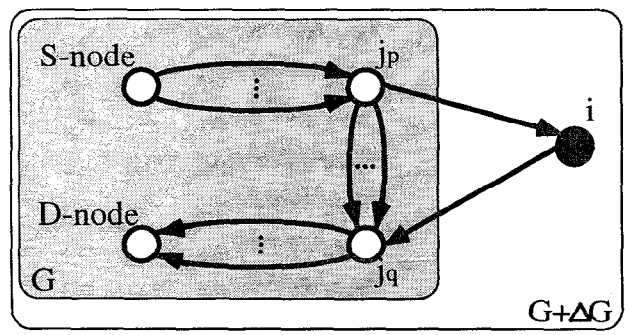

Figure 8: Concept of PE added

5.3 Example

Consider the protocol specification in Figure 7 and the topology change such that one PE (let it be PE_6) is added. In this case, there exist two communication channels between PE_3 and PE_6 and between PE_5 and PE_6.

In the proposed method, only **PE-SPEC**s that are related to newly added channels are modified. In this

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE

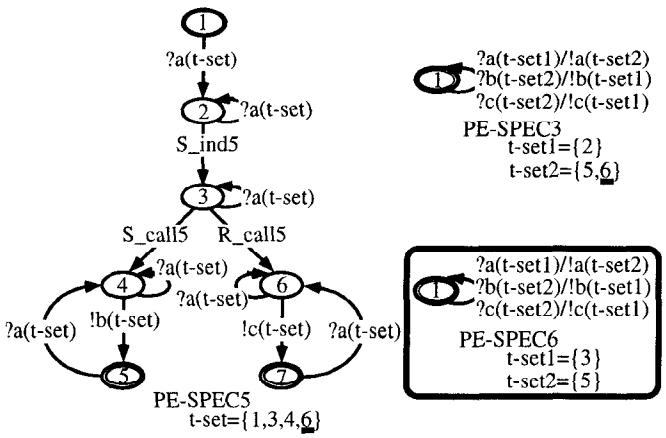

Figure 9: Modified protocol specification

example, the protocol entity specifications for PE_3 and PE_5 are modified. In **PE-SPEC**s, PE_5 is a D-node of paths for delivery of message **a** and it is relayed through PE_3. Then, we have newly a possibility that message **a** is relayed through PE_3 and PE_6 and is finally received by PE_5.

On the other hand, PE_5 is an S-node of paths for delivery of messages **b** and **c**, and they are relayed through PE_3. Then, we have newly possibilities that messages **b** and **c** are relayed through PE_6, and then are received by PE_3. Therefore, $t\text{-}set2 = \{5\}$ in **PE-SPEC**3 is updated to $t\text{-}set2 = \{5,6\}$, and $t\text{-}set = \{1,3,4\}$ in **PE-SPEC**5 is updated to $t\text{-}set = \{1,3,4,6\}$. **PE-SPEC**6 is similarly modified for relaying message **a** from PE_3 to PE_5 and messages **b** and **c** from PE_5 to PE_3.

Figure 9 shows only the **PE-SPEC**s that are modified for adding PE_6. In Figure 9, the rectangle surrounded by bold lines and the values specified by underlines represent the changes needed for updating **PE-SPEC**s in the modified protocol specification. It is clear that the number of changes needed for updating **PE-SPEC**s are very small.

6 Conclusion

In this paper, we have proposed a new synthesis method which generates a fault-tolerant and flexible multipath routing protocol from a given service specification. The proposed method enables derivation of such a fault-tolerant protocol specification that messages are rerouted at the source node and delivered to the destination node even when a communication path fails. So, the proposed design method enables the efficient production of reliable fault-tolerant routing protocol specification at a lower cost.

Furthermore, for the given network topology changes, only protocol entity specifications corresponding to the changes need to be modified in the obtained protocol. This is useful for routing protocol for large number of nodes.

Although we have assumed that at most one failure occurs on the network, we can relax this assumption and we are currently trying to extend the proposed method for attaining it . Additionally, we are also developing a computer-assisted tool to evaluate the effectiveness of the proposed method.

References

[1] Chu, P.M. and Liu, M.T., "Synthesizing protocol specifications from service specifications in the FSM model," *Proc. Computer Networking Symp.*, pp.173-182, April 1988.

[2] Chu, P.M. and Liu, M.T., "Protocol synthesis in a state transition model," *Proc. International Computer Software and Applications Conference (COMPSAC'88)*, pp.505-512, October 1988.

[3] Dolev S., and Welch, J. L., "Crash resilient communication in dynamic networks," *IEEE Trans. on Computers*, Vol.46, No.1, pp.14-26, Jan. 1997.

[4] Herzberg, A., "Connection-based communication in dynamic networks," *Proc. 11th Ann. ACM Symp. Principles of Distributed Computing* pp.13-24, 1992.

[5] Hopcroft, J.E. and Ullman, J.D., *"Introduction to Automata Theory, Language, and Computation"*, Addison-Wesley, 1979.

[6] Kakuda, Y., Nakamura, M., and Kikuno, T., "Automated synthesis of protocol specifications with parallely executable multiple primitives," *IEICE Trans. on Fundamentals*, Vol.E77-A, No.10, pp.1634-1645, October 1994.

[7] Leewen J. V., *"HandBook of Theoretical Science"*, Elsevier Science Publishers B. V, 1990.

[8] Liu, M. T.,"Protocol Engineering," Advances in Computers, 29, pp.79-195, Academic, 1989.

[9] Probert R. L., and Saleh K., "Synthesis of communication protocols: survey and assessment," *IEEE Trans. on Computers*, Vol 40, No. 4, pp.468-476, April 1991.

[10] Saadawi T. N., Ammar M.H., and Hakeem A.E., *"Fundamentals of Telecommunication Networks"*, John Wiley & Sons, 1994.

[11] Saleh, K., "Automatic synthesis of protocol specifications from service specifications," *Proc. Int'l Phoenix Conference on Computers and Communications (IPCCC'91)*, pp.615-621, March 1991.

[12] Singh G., and Sammeta M., "On the construction of multiphase communication protocols" *Proc. International Conference on Network Protocols (ICNP'94)*, pp.151-158, October 1994.

Proceedings of the 3rd International Conference on Engineering of Complex Computer Systems (ICECCS'97)
0-8186-8126-8/97 $10.00 © 1997 IEEE