A Synthesis Method for Fault-Tolerant Multicast Routing Protocol

Kenji Ishida and Kazuhiro Amano

Faculty of Information Sciences, Hiroshima City University,
3-4-1 Asaminami, Hiroshima 731-3194, 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 whenever 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 and a network topology, and a tree topology rate.

1 Introduction

There are real-time tasks in which the computer system must respond to input stimuli on a time delay. In such a task, it is often required by a number of time-critical services. Such distributed computing will need multicast communication services. 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 topology rate. In this case, messages are replicated by copy and send on the tree. The protocol is defined to be fault-tolerant if messages can be retransmitted along with a message loss occurs. Fault-tolerant becomes an important characteristic to ensure quality of service multicast communication services [1], [2].

Design of practical multicast routing protocol is complex and difficult due to complicated requirements of fault-tolerance and service hierarchy of messages in the copy and send. For such a difficult and complex protocol design, the protocol synthesis [5], [7] is regarded as one of the most promising solutions, which automatically derives the protocol specification from specification on the tree. In this paper, a synthesis of multicast routing protocol is defined as generation of a routing protocol specification from a routing service specification.

Up to this paper time, various protocol synthesis methods have been proposed [3], [6], [7], [8]. However, none of them was for multicast routing protocol in which the behavior of copy and send must be taken into consideration.

This paper proposes a new synthesis method for multicast routing protocol in which behavior of a copy and send in the protocol.
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 the synthesis problem for multicast routing and propose 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 Communicating 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’ viewpoint. The nodes are also called protocol entities which are denoted by PEs in this paper. 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: 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 exist 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 tree-shaped route from a $S$-Node to a $D$-Node. 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 a $D$-Node. First $PE_1$ sends a message $a$ to $PE_2$ (a copy node). Then, $PE_2$ receives $a$ and sends $a_1$ and $a_2$ to $PE_3$ and $PE_5$, respectively. When $a$ and $a_2$ are copies of the message $a$. Then, $PE_3$ and $PE_5$ receive $a_1$ and $a_2$, respectively. Then, $PE_3$ and $PE_5$ send ACK or NAK 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 NAK message to $PE_1$.  

![Fig. 1. A topology graph and tree-shaped route](image-url)
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 nodes." Another is a set of service specifications between a "copy node" and "D-nodes." 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 those 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 and PE 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 \( \varepsilon \) 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 \( v = T_s(u, p, u, v \in \sum_s) \).
- \( \sigma \) is an initial service state.

A state \( u \in S_s \) is called a final state if there is no outgoing transition \((u, p, v)\) for any \( p \) and \( v \). If there is an outgoing transition \((u, p, v)\) from a service state, one of such transitions is chosen and executed. We call this FSM a service specification S-SPEC. A 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 \( OUT(S) = \{ p | \exists (u, p, v) \} \), 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 positive or negative acknowledgment from the destination node to the source node. For example, after \( user_1 \) sends \( S_{req} \) to \( PE_3 \) through SAP_1, \( user_2 \) receives \( S_{req} \) from \( PE_3 \) through SAP_2 in this order. In case of \( user_2 \) sends \( S\_call \) (ACK) through SAP_1, \( user_1 \) receives \( S\_conf \) from \( PE_1 \) through SAP_1. In case of \( user_2 \) sends \( R\_call \) (NACK) through SAP_2, \( user_1 \) receives \( S\_conf \) 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.
Definition 3. A state $u$ is called a fork state if there are several outgoing transitions $(u, p_1, v_1), (u, p_2, v_2), \ldots, (u, p_k, v_k)$ 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 $(v_1, p_1, u), (v_2, p_2, u), \ldots, (v_k, p_k, u)$ in the S-SPEC. All 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 executes the other incoming transitions. All 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 Pseudo Code Specification

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

Definition 5. If a message $e$ is transmitted to $PE_j$, then it is denoted by a transmission event $le(j)$. On the other hand, if a message $e$ is received by $PE_j$, then it is denoted by a reception event $re(j).
Definition 6. A protocol entity specification is modeled by an FSM \( P_i = \langle S_{ip}, T_{ip}, \sigma_{ip} \rangle \) 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. \( \sum_{ip} = \{ \delta | \delta \in \sum_s, \sigma p(\delta) = i \} \cup MEX_i \cup \{ T, O \} \cup \{ \varepsilon \} \), where \( \sum_s \) is a set of primitives in Definition 2, and \( MEX_i \) is a set of events which are transmitted to \( PE_{i1}, PE_{i2}, \ldots, PE_{ik} \) or received by \( PE_{i1}, PE_{i2}, \ldots, PE_{ik} \) and \( T, O \) is a timeout event that occurs when a predetermined time elapses. \( \varepsilon \) 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-SHC \(_i\). 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-SHC \(_i \) moves to state \( u \), counting timer starts. Only when a current state of PE-SHC \(_i \) is state \( u \) and the predetermined time elapsed, the state of PE-SHC \(_i \) moves to state \( v \). A transition \( -p/q \) with \( p, q \in T_{ip} \) denotes a successive execution of transitions \( p \) and \( q \).
Example 4. Figure 4 shows an example of PE-SPEC for PE1 in Fig.1. In this figure, a circle denotes a protocol state, and an arrow between states denotes a transition. For example, $\text{S}_\text{req1}/a(2)$ implies that user1 sends $\text{S}_\text{req1}$ to $PE_1$ through SAP1, 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 $S\_\text{call12}(\text{ACK})$ and $c$ corresponds to $R\_\text{call12}(\text{NACK})$. 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 $P_i = \langle S_i, \sum_{ip}, T_{ip}, \sigma_{ip} \rangle$. 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 $(u, e(j), v) / \mathbb{E}_{ip}$ for any $u, v, e$.
- If $(i, j) \in E'$, then $(u, ?e(j), v) / \mathbb{E}_{ip}$ 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. These applications use multicast transmission for resource discovery or multimedia conferences [4]. These application require fault-tolerant multicast routing.

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

Table 1. Transition synthesis rules

<table>
<thead>
<tr>
<th>Input</th>
<th>Condition</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Out(s2)={i}</td>
<td>x=Out(s1)</td>
</tr>
<tr>
<td>B1</td>
<td>x=Out(s2) PS-SPEC j (j ≠ i)</td>
<td>ε=Out(s2)</td>
</tr>
<tr>
<td>A2</td>
<td>Out(s2) ≠ {i}</td>
<td>y=Out(s1)</td>
</tr>
<tr>
<td>B2</td>
<td>x=Out(s2) PS-SPEC j (j ≠ i)</td>
<td>x=Out(s2)</td>
</tr>
</tbody>
</table>

which satisfies the following conditions:

1. Unspecified exceptions never occur in $P$. 
2. Even if a message loss occurs, the execution order of service primitives defined by $S_i$ obeys $P$. 
3. $P$ obeys the channel restriction.

Non-existence of unspecified exceptions condition and keeping execution order of service primitives in Condition 2 are ordinary conditions for protocol synthesis. Meanwhile the channel restriction based on $S_i$ condition and multi-camera communication channels make our discussion.

3.2 Outline of Synthesis Method

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

Step 1: Obtain projected service specification by applying the projection to the given service specification $SPEC$. In these projected service specifications, the service primitive associated with $SAP_i$ is represented by $SPEC_i$, which is obtained from $SPEC_i$ by substituting the transition associated with $SAP_i$ by $ε$.

Step 2: Construct $SPEC_i$ by applying transition synthesis rules shown in Table 1 to $SPEC_i$ obtained in Step 1.

Step 3: Incorporate the capability of time out and retransmission into $SPEC_i$. Next, introduce the channel restriction $PE SPEC_i$. Next, remove $ε$ transition from $SPEC_i$ by the algorithm [3]. Finally, based on the result of $SPEC_i$, the relation among the $SPEC_i$s, we obtain the final protocol specification.

3.3 Example of Protocol Synthesis Method

In this subsection, apply the synthesis method to a typical example. Consider a service specification of $SPEC_i$ shown in Fig. 2 and Fig. 3 and a topology graph $G$ shown in Fig. 1.
At Step 1, service primitives are projected to PS-SPEC1, PS-SPEC2, PS-SPEC3, PS-SPEC4, PS-SPEC5. For S-SPEC (a) in Fig. 2, the result of step 1 is shown in Fig. 5. Similarly, the other S-SPECi (1 ≤ i ≤ 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-SPEC1 are obtained from PS-SPEC1. A S-SPEC (a) in Fig. 5, each PE-SPEC1 for (a) in Fig. 6 shows the result of step 2. Similarly, the other PE-SPECi are obtained from PE-SPEC (b) and (c).

At Step 3, firstly construct each HS-SPEC that incorporates the capability of time-out and retransmission into HS-SPECi constructed at Step 2. For example, two time-out transitions TOi are added to HS-SPEC2 in Fig. 7 and several self-loops are added to HS-SPECi (2 ≤ i ≤ 5) in Fig. 7. Next, introduce the channel restriction (Condition 3) to HS-SPECi. Intuitively, the channel restriction translates HS-SPECi from an end-to-end communication into a link-to-link communication. For example, HS-SPEC4 in Fig. 6 executes nothing. However,
Fig. 7. Final protocol specification

3.4 Fault-tolerance of Routing Protocol

In this subsection we discuss how the protocolspecificationscanbyour method realisefault-tolerant.Consider the protocol specifications in Fig. 7 and assume that messages between PE2 and PE5 in Fig. occurs.

For example, d'2 delivers a message through the channel between PE2 and PE3. But the message gets lost because the transition did not happen.
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-SMIC2 in Fig. 7). It is clear that for the transition of 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 sec for some real-time applications to 1 sec or less for interactive communication terminal. In this paper, a timeout 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 or 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

3. Lipsett J.M., Ullman J.D. Introduction to Automata Theory, Languages, and Computation /Allyn & Bacon (1973)

This article was produced using the $\LaTeX$ macro package with $\LaTeX$ style.