Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp410-417)

Automatic Synthesis of Timed Protocol Specifications from Service Specifications

JEHAD AL DALLAL
Department of Information Sciences
Kuwait University
P.O. Box 5969, Safat 13060
KUWAIT

Abstract: - Several methods have been proposed for synthesizing computer communication protocol specifications from service specifications. In real time applications, the time required to execute the events can be crucial and has to be considered. Some of the protocol synthesis methods do not consider timing constraints and, therefore, cannot be used in real time applications. In this paper, the assignment of the timing constraints to the service specification is discussed. In addition, an automatic method for synthesizing protocol specifications is extended to consider timing constraints given in the service specification. Both the service and protocol specifications are modeled using Timed Finite State Machines (TFSMs). The resulting synthesized protocol is guaranteed to conform to the timing constraints given in the service specification.

Key-Words: - protocol synthesis, protocol specification, service specification, timing constraints, TFSM.

1 Introduction

A protocol can be defined as an agreement on the exchange of information between communicating entities. A full protocol definition defines a precise format for valid messages (a syntax), procedure rules for the data exchange (a grammar), and a vocabulary of valid messages that can be exchanged, with the meaning (semantics).

In protocol design, interacting entities are constructed to provide a set of specified services to the service users. While designing a communication protocol, semantic and syntactic errors may exit. Semantic design errors cause the provision of incorrect services to the distributed protocol users. Syntactic design errors cause the protocol to deadlock.

A communication system is most conveniently structured in layers. The Service Access Point (SAP) is the only place where a layer can communicate with its surrounding layers or service users. The layer can have several SAPs. The communication between the layer and its surrounding is performed using Service Primitives (SPs). The SP identifies the type of event and the SAP at which it occurs.

From user's viewpoint (high level of abstraction), the layer is a black box where only interactions, identified by the SPs, with the user are visible. The specification of the service provided by the layer is defined by the ordering of the visible SPs and the timing requirements between the SP occurrences. This specification is called Service Specification (S-SPEC). At a refined level of abstraction, the service provided by the layer is performed using a number of cooperating protocol entities. These protocol entities exchange protocol messages through a communication medium. The protocol specification (P-SPEC) prescribes the exchange of messages between the protocol entities. Figure 1 shows the two abstraction levels of a communication layer. Both S-SPEC and P-SPEC can be modeled using Communicating Finite State Machines (CFSMs).

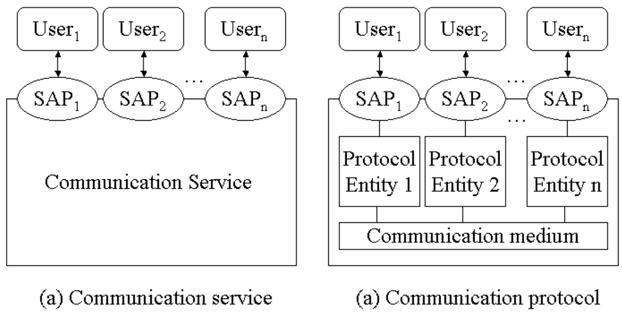

Figure 1. The communication service and protocol concepts

Protocol specifications are much complex than service specifications because of their refined nature. Therefore, it is quite natural to start the protocol design process from a complete and unambiguous service specification. The construction of a protocol specification from a given service specification is called a *protocol synthesis*. Protocol synthesis is relatively an easy and time-saving task. That is, instead of applying a sequence of design, analysis, error detection and correction iteratively until the design becomes error-free, protocol synthesis approach does not require any further validation. The synthesis approach is used to construct or complete a partially specified protocol design such that the interactions between the constructed or completed protocol entities proceed without encountering any logical error and ideally provide the specified service. In addition, the syntactic correctness of the synthesized protocol is often a direct byproduct of the synthesis method [1]. Several protocol synthesis methods have appeared in the literature such as [2,3,4,5,6,7,8,9,10,11]. Most of these methods do not consider the timing requirements given in the service specification and, therefore, cannot be used for real time applications.

Saleh and Probert [2] have proposed an automatic synthesis method of CFSM-modeled protocol specification starting from the service specification without considering the timing constraints. In this paper, the assignment of the timing constraints to the service specification is discussed. In addition, Saleh and Propert method is extended to synthesize protocol specifications from service specifications containing timing requirements. The resulting protocol specification is proved to conform to the timing constraints provided in the service specification.

The paper is organized as follows. In Section 2, the model used for the service and protocol specifications is defined. The related research is overviewed in Section 3. In Section 4 the service specification time assignment is discussed and the timed protocol synthesis method and a small example are introduced. The correctness of the synthesis method is proved in Section 5. Finally, Section 6 provides conclusions and discussion of future work.

2 Model Definition

In this paper, both the service and protocol specifications are modeled using Finite State Machines (FSMs). In general, FSMs consist of states and transitions. In this paper, the FSM is extended by associating time constraints with the transitions. The extended model is called Timed Finite State Machine (TFSM). In this section, the TFSM is formally defined for the specification of the services and protocols in the context of the layered communication system introduced in Section 1.

2.1 Service specification model

The service specification described in TFSM defines sequences of primitives exchanged between users and processes through the service access points.

Definition 1: A service specification S-SPEC is modeled by a TFSM denoted by a tuple (S_s,T_s,σ) where:

1. S_s is a non-empty finite set of service states.
2. T_s is a finite set of transitions such that each transition $t \in T_s$ is a 4-tuple $<$*head(t)*, *tail(t)*, *SP*, $[min_t,max_t]>$ where:

a. *head(t)* and *tail(t)* are respectively the head and the tail states of *t*.

b. *SP* is the service primitive that defines the service event, its type, and the index of the SAP through which the *SP* passes. There are two types of service events ↑ and ↓. An SP of type ↑ is an SP directed upward from the protocol entity to the SAP. The SP of type ↓ is an SP directed downward from the service user at a SAP to a protocol entity.

c. $[min_t,max_t]$ is the time interval associated with *t* such that the transition *t* can be executed only within the time T since *head(t)* is visited, where $min_t \le T \le max_t$.

3. $\sigma \in S_s$ is the initial service state.

Figure 2 shows a S-SPEC example. In this example, S_s={s1, s2, s3, s4}, T_s= {<s1,s2,A_1↓,[1,3]>, <s2,s3, B_2↓,[1,4]>, <s2,s4,C_3↓,[2,3]>, <s3,s4,D_1↓,[1,2]>, <s4,s1,E_2↓,[1,2]>}, and σ={s1}.

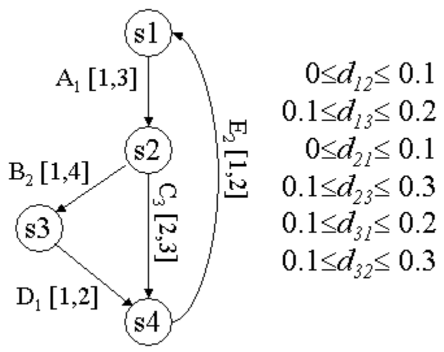

Figure 2. A service specification example

Definition 2: A projected service specification PS-SPEC$_i$ is the projection of the S-SPEC onto SAP$_i$. The PS-SPEC$_i$ is modeled by a TFSM denoted by a tuple (S_s', T_s', σ') where:

1. $S_s' = S_s$
2. T_s' = {<*head(t)*, *tail(t)*, *SP*, $[min_t,max_t]$>| $t \in T_s$ and SAP(*SP*)=*i*}∪ {<*head(t)*, *tail(t)*, ε, ε >| $t \in T_s$ and SAP(*SP*)≠ *i*}
3. $\sigma' = \sigma$

Figure 3 shows the projected service specifications for the S-SPEC given in Figure 2. In PS-SPEC$_1$, S_s'={s1, s2, s3, s4}, T_s'= {<s1,s2,A↓,[1,3]>, <s2,s3, ε,ε>, <s2,s4,ε,ε>, <s3,s4,D↓,[1,2]>, <s4,s1,ε,ε>}, and σ={s1}.

2.2 Protocol specification model

The protocol specification consists of the specifications of the protocol entities that cooperate to provide the service described in the service specification.

Definition 3: The protocol entity specification PE-SPEC$_i$ is also modeled by a TFSM denoted by a tuple (S_{pi},T_{pi},σ_{pi}) where:

1. S_{pi} is a non-empty finite set of states of protocol entity *i*.
2. T_{pi} is a finite set of transitions such that each transition $t \in T_{pi}$ is a 4-tuple <*head(t)*, *tail(t)*, E_i, [min_t,max_t]> where:

a. *head(t)* and *tail(t)* are respectively the head and the tail states of *t*.

b. E_i is a protocol event that can be either (1) an SP that passes through SAP$_i$, (2) an event message *E* sent from PE$_i$ denoted by !e_i, or (3) an event message *E* received by PE$_i$ denoted by ?e_i.

c. [min_t,max_t] is the time interval associated with *t* such that the transition *t* can be executed only within the time T since *head(t)* is visited, where $min_t \le T \le max_t$. The time interval is associated with only the transition that has the first type of E_i. The time required for sending an event from PE$_i$ and receiving the event by PE$_j$ (i.e., protocol events of Types 2 and 3) is controlled by the delay of the channel between the two protocol entities. This delay is considered -as will be illustrated in Section 4- when computing the time interval for the SP (i.e., the protocol event of Type 1). Therefore, no time interval is associated with transitions that have events of Types 2 and 3.

3. $\sigma_{pi} \in S_{pi}$ is the initial protocol state.

Figure 5 shows three PE-SPEC examples. For PE-SPEC$_1$, S_{pi}={s1, s2, s3}, T_{pi}= {<s1,s2,A/!$a_{2,3}$, [1,2.8]>, <s2,s3,?b_1,ε>, <s3,s2,D/!d_2,[1,1.9]>, <s2, s1,?e_2,ε>}, and σ_{pi}={s1}. In this work, we assume that the communication medium between the protocol entities is reliable and the messages are delivered in the first-in-first-out (FIFO) order. Each channel between two protocol entities PE$_i$ and PE$_j$ has a delay d_{ij} such that $\min(d_{ij}) \le d_{ij} \le \max(d_{ij})$, where $\min(d_{ij})$ and $\max(d_{ij})$ are respectively the minimum and maximum delay of the channel from PE$_i$ to PE$_j$.

3 Related Research

In this section, an overview of other related research is provided and the basic service-oriented synthesis method introduced in [2] is briefly described.

3.1. Other Related Research

Two approaches are used in designing communication protocols: analysis and synthesis. In the analysis approach, a sequence of design, analysis, error detection and correction is applied iteratively to produce error-free design. In the synthesis approach, the protocol design is constructed or completed in such a way that no further validation is needed. Some protocol synthesis methods start the derivation process from a complete service specification [2,3,4,5,6,7,8,9,10,11,12] and others do not [13,14]. The protocol synthesis methods can be further classified according to the used models. The used models include finite state machines [2,4,9,10], Petri-nets [5,11,12], and LOTOS-like [3,6,7,8].

Some of the service-oriented protocol synthesis methods consider the timing requirements given in the service specification [7,8,9,11] and others do not [2,3,4,5,6,10]. The method of dealing with timing constraints provided in the service specifications in [7,8, and 11] cannot be directly applied in this paper because a different model is used (i.e., Petri-nets and LOTOS-like models). In [8], the channel delay is assumed negligible while in [7], the minimum channel delay is assumed to be always zero. In [9], the timing constraints provided in the service specifications that have concurrency behavior are considered. The paper assumes that only the upper bound of delay for each channel is given and the lower bound is assumed to be always zero as in [7]. In this paper, the timing constraints provided in the service specifications that have sequential behavior are considered and the lower bound of the channels is generalized to be any nonnegative value.

3.2. The Basic Synthesis Method

The synthesis method introduced in [2] uses FSM to model both service and protocol specifications. The models are similar to the models introduced in Section 2 except for the time interval associated with the transitions. The timing constraints are not considered in the basic synthesis method.

To synthesize the protocol specification from the service specification, three steps are followed.

1. Project the service specification S-SPEC onto each SAP to obtain the PS-SPECs defined in

Definition 2.

2. Apply the transition synthesis rules to each transition in the PS-SPECs to obtain the PE-SPECs. The transition synthesis rules are the same as the rules given in Table 1 but with no time intervals.

3. Remove ε-cycles and ε-transitions by using algorithms described in [15] to obtain the reduced PE-SPECs.

Ignoring the time intervals given in Figures 2, 3, 4, and 5, Figures 3, 4, and 5 show the PS-SPECs, PE-SPECs, and reduced PE-SPECs after applying Steps 1, 2, and 3, respectively, for the S-SPEC given in Figure 2.

4 Timed Protocol Synthesis Method

To synthesize timed protocol specifications, the service specification has to be provided with time constraints associated with the transitions of the TFSM. In this section, the time assignment to the S-SPEC transitions is discussed and the synthesis method for the timed protocol specification is introduced. Finally, a small example is illustrated.

4.1 Service specification time assignment

The assignment of the service specification time constraints is performed during the S-SPEC design process. These time constraints are assigned as time intervals associated with the transitions of the TFSM that models the S-SPEC. The time interval $[min_t,max_t]$ means that the transition t can be executed only within the time T since the source state of t is visited, where $min_t \leq T \leq max_t$. The time T includes the waiting time T_w since the source state is visited. If the SP associated with the transition is to be sent from one Protocol Entity (PE) to another, the time T includes also the time required for sending the SP from the source PE and receiving the SP by the destination PE. The time for sending and receiving an SP from PE_i to PE_j is the delay d_{ij} of the channel between the two PEs. Therefore, $min_t=\min(T_w)+\min(d_{ij})$ and, consequently, min_t has to be greater than or equal to $\min(d_{ij})$. Similarly, $max_t=\max(T_w)+\max(d_{ij})$ and, consequently, max_t has to be greater than or equal to $\max(d_{ij})$. In addition, the min_t and max_t have to be assigned such that $\max(T_w)\geq\min(T_w)$. In other words, $max_t-\max(d_{ij})\geq min_t-\min(d_{ij})$. Thus, $max_t\geq min_t+(\max(d_{ij})-\min(d_{ij}))$. In some cases, an SP associated with a transition can be sent to more than one PE (e.g., in Figure 2, A_1 is sent to PE_2 and PE_3). Let X be a set of the protocol entities that can receive the SP. Generally, if an SP associated with a transition t can be sent from PE_i to more than one PE such that each PE∈X, the time interval associated with t has to be assigned such that $\forall j\in X$, $min_t\geq\min(d_{ij})$ and $max_t\geq\max(d_{ij})$. This means that $min_t\geq\text{maximum}_{\forall j\in X}(\min(d_{ij}))$ and $max_t\geq\text{maximum}_{\forall j\in X}(\max(d_{ij}))$. Similarly, $\forall j\in X$, $max_t\geq min_t+(\max(d_{ij})-\min(d_{ij}))$. This means that $max_t\geq min_t+\text{maximum}_{\forall j\in X}(\max(d_{ij})-\min(d_{ij}))$.

For example, in Figure 2, the service primitive A_1 is sent to PE_2 and PE_3. You can notice that the conditions $max_t\geq\text{maximum}(\max(d_{12}),\max(d_{13}))$ (i.e., 3>maximum(0.1,0.2)), $min_t\geq\text{maximum}(\min(d_{12}),\min(d_{13}))$ (i.e., 1>maximum(0,0.1)), and $max_t\geq min_t+\text{maximum}((\max(d_{12})-\min(d_{12})),(\max(d_{13})-\min(d_{13})))$ (i.e., 3>1+maximum((0.1-0),(0.2-0.1)) are satisfied.

4.2. Synthesis of timed protocol specifications

An automatic synthesis method for the protocol entities from a service specification is introduced in [2] and summarized in Section 3. In this section, the synthesis method is extended to consider the timing constraints provided in the service specification.

To consider the timing constraints, the first two steps of the basic method are extended. Then the third step is applied as-is.

Step 1 Extension

In this first step of the basic synthesis method, the service specification S-SPEC is projected onto each SAP to obtain the PS-SPECs. The PS-SPEC obtained by the projection of the S-SPEC onto SAP_i includes the same states and transitions of the S-SPEC. The only difference is in the labels of the transitions associated with the events that do not pass through SAP_i. These events are substituted by ε-events. In the basic synthesis method, the transitions of the PS-SPECs are not associated with time intervals because the S-SPEC does not include them. In the extended synthesis method, the transitions of the PS-SPECs are associated with the same time intervals associated with the transitions of the S-SPEC. The PS-SPEC transitions associated with ε-events are not assigned to time intervals. Figure 3 shows the PS-SPECs derived from the S-SPEC given in Figure 2.

Step 2 Extension

In the second step of the basic synthesis method, a set of transition rules are applied to each transition (ε or SP-labeled) in the SP-SPECs to obtain the protocol entities. In the extended synthesis method, these rules are extended to consider the time intervals associated with the transitions of the PS-SPEC. The extended rules and the conditions for

Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp410-417)

their applications are summarized in Table 1. In this table, OUT(s) means the SAPs at which the events associated with the outgoing transitions from state S pass through. For example, in Figure 2, OUT(s2)={2,3} because the service primitive B passes through SAP_2 and the service primitive C passes through SAP_3. The intuition for these extensions are given below.

a. Transition labeled by an SP in $PS\text{-}SPEC_i$:

Rule a.1: This rule implies that the flow of control needs not be transferred to another protocol entity or service user. Therefore, no channel delays are to be considered. In this case, the same time interval is considered without changing.

Rule a.2: In this case, the transition is taking back the service to its initial state and, therefore, a synchronization message is sent to all other PEs. Thus, the channel delays between the PE_i and all other PEs have to be considered. In this case, the maximum and the minimum channel delays among the considered ones are respectively subtracted from max_t and min_t of the transition to obtain the new max_t and min_t values.

Rule a.3: In this case, the SP is sent to a service user not to another PE. Therefore, no channel delays are to be considered. In this case, the same time interval is considered without changing.

Rule a.4: In this case, the SP is originating from the service user at SAP_i. After the occurrence of this SP, other SPs are observed at other SAPs. A synchronization message is sent from PE_i to the other corresponding PEs. Therefore, the channel delays between the PE_i and the other corresponding PEs have to be considered as illustrated in Rule a.2.

Rule a.5: The intuition of this rule is similar to Rule a.3.

Conditions \ Transition type	$s1 \xrightarrow{E_i\,[min_t, max_t]} s2$	$s1 \xrightarrow{\varepsilon} s2$
s2 is not an initial state and OUT(s2)= SAP_i	a.1: $s1 \xrightarrow{E\,[min_t, max_t]} s2$	b.1: $s1 \xrightarrow{\varepsilon} s2$
s2 is an initial state and E is of type ↓	a.2: $s1 \xrightarrow[X=\text{all SAPs}-SAP_i]{E/!e_X\,[min_t-\min_{\forall j\in X}(\min(d_{ij})),\ max_t-\max_{\forall j\in X}(\max(d_{ij}))]} s2$	b.2: $s1 \xrightarrow{?e_i} s2$
s2 is an initial state and E is of type ↑	a.3: $s1 \xrightarrow{E\,[min_t, max_t]} s2$	b.3: $s1 \xrightarrow{\varepsilon} s2$
s2 is not an initial state, OUT(s2)≠SAP_i and E is of type ↓	a.4: $s1 \xrightarrow[X=OUT(s2)-SAP_i]{E/!e_X\,[min_t-\min_{\forall j\in X}(\min(d_{ij})),\ max_t-\max_{\forall j\in X}(\max(d_{ij}))]} s2$	b.4: $s1 \xrightarrow{?e_i} s2$
s2 is not an initial state, OUT(s2)≠SAP_i and E is of type ↑	a.5: $s1 \xrightarrow{E\,[min_t, max_t]} s2$	b.5: $s1 \xrightarrow{\varepsilon} s2$

Table 1. Summary of the transition synthesis rules and the conditions for their application

b. Corresponding transition labeled by ε in another PS-SPEC

The transition associated with ε-event is either remains the same (Rules b.1, b.3, and b.5) or is associated with a receiving message for the synchronization message sent by PE_i. The transition associated with an ε-event is not assigned a time interval and, therefore, no timing constraints are to be considered. In addition, The transition associated with a receiving message is not assigned a time interval because the time required to execute this transition is part of the channel delay already considered in Rules a.2 and a.4.

4.3 Example

Figure 2 shows a S-SPEC example. Figure 3 shows the three PS-SPECs obtained by applying Step 1 of the extended synthesis method. Finally, Figures 4 and 5 show the three PE-SPECs resulting from applying Steps 2 and 3 of the extended and basic synthesis methods, respectively. In PE_1, the transition associated with the service primitive A has the time interval [1-min(min(d_{12}),min(d_{13})),3-max (max(d_{12}),max(d_{13}))] and the transition associated with the service primitive D has the time interval [1-min(d_{12}),2-max(d_{12})]. In PE_2, the transition associated with the service primitive E has the time interval [1-min(min(d_{21}),min(d_{23})),2-max(max(d_{21}), max(d_{23}))] and the transition associated with the service primitive B has the time interval [1-min(d_{21}) ,4-max(d_{21})]. Finally, In PE_3, the transition associated with the service primitive C has the time interval [2-min(d_{32}),3-max(d_{32})].

5 Proof of correctness

Proving the correctness of the synthesis method requires proving that the synthesis method is syntactically and semantically correct. This proof is provided in [2] but without timing constraints. Therefore, to complete the proof, we prove here that the time assignments to the transitions of the PEs as a result of applying the extended synthesis method conform to the time constraints assigned to the transitions of the S-SPEC.

Lemma 1. In the PEs, the time T required for executing an SP is $min_{tp} \le T \le max_{tp}$ such that the time interval [min_t,max_t] is associated with the corresponding transition in the S-SPEC and $min_t \le min_{tp} \le T \le max_{tp} \le max_t$.

Proof: An SP executed in a PE is either (1) not sent to another PE (Rule a.1), (2) sent to a service user

Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp410-417)

(Rules a.3 and a.5), or (3) sent to one or more other PEs (Rules a.2 and a.4). In the first two cases, the SP is not sent to another PE and, therefore, no channel delays are to be considered. As a result, in these two cases, the time required to execute the SP in the PE is the same as the time associated with the corresponding transition in the S-SPEC (i.e., $(min_t = min_{tp}) \leq T \leq (max_{tp} = max_t)$).

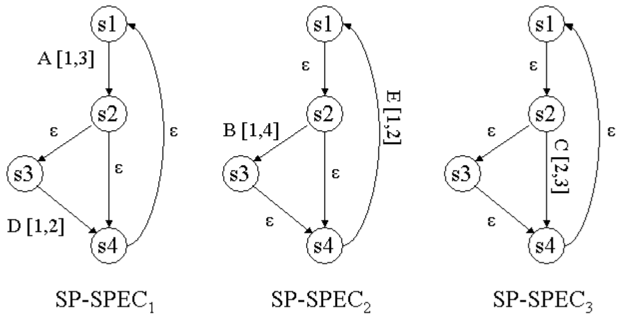

Figure 3. The PS-SPECs obtained by applying Step 1 of the extended synthesis method.

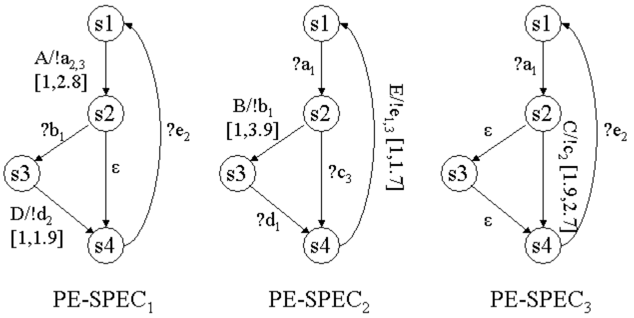

Figure 4. The PE-SPECs obtained by applying Step 2 of the extended synthesis method.

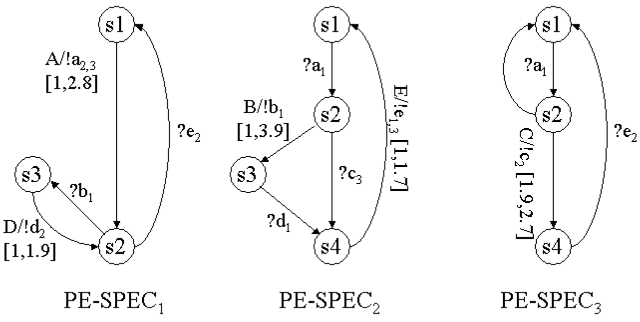

Figure 5. The PE-SPECs obtained by applying Step 3 of the basic synthesis method.

For the third case, the SP is either sent to another PE or sent to more than one other PEs. If the SP is sent to another PE, the time required to execute the SP in the PE is the waiting time since the source state is visited and the channel delay d_{ij}. The waiting time is the time associated with the PE transition labeled by SP. As given in Rule a.2, this time is T such that $min_t\text{-min}(d_{ij}) \leq T \leq max_t\text{-max}(d_{ij})$. As a result, the time required to execute the SP in the PE (i.e., waiting time + channel delay) is T such that $min_t\text{-min}(d_{ij})+d_{ij} \leq T \leq max_t\text{-max}(d_{ij})+d_{ij}$. This means that $min_{tp}= min_t\text{-min}(d_{ij})+d_{ij}$ and $max_{tp}= max_t\text{-max}(d_{ij})+ d_{ij}$. Since $min_t\text{-min}(d_{ij})+d_{ij}$ and $\text{min}(d_{ij}) \leq d_{ij}$, then $min_t\text{-min}(d_{ij})+\text{min}(d_{ij}) \leq min_t\text{-min}(d_{ij})+d_{ij} \leq T$. As a result, $min_t \leq min_{tp} \leq T$. Similarly, since $T \leq max_t\text{-max}(d_{ij})+d_{ij}$ and $d_{ij} \leq \text{max}(d_{ij})$ then $T \leq max_t\text{-max}(d_{ij})+ d_{ij} \leq max_t\text{-max}(d_{ij})+\text{max}(d_{ij})$. Therefore, $T \leq max_{tp} \leq max_t$. As a result, in this case, in the PEs, the time T required for executing an SP is T such that $min_t \leq min_{tp} \leq T \leq max_{tp} \leq max_t$.

The last case is when the SP is sent from one PE to more than one other PEs. In this case, the waiting time associated with the transition labeled by SP, as given in Rule a.2, is T such that $min_t\text{-minimum}_{\forall j \in X}(\text{min}(d_{ij})) \leq T \leq max_t$ - $\text{maximum}_{\forall j \in X}(\text{max}(d_{ij}))$ where X is the set of the protocol entities that can receive the SP. When considering the channel delays, the minimum time T required to execute the SP is calculated such that $min_t\text{-minimum}_{\forall j \in X}(\text{min}(d_{ij}))+\text{minimum}_{\forall j \in X}(d_{ij}) \leq T$. Since $\text{minimum}_{\forall j \in X}(\text{min}(d_{ij})) \leq \text{minimum}_{\forall j \in X}(d_{ij})$, then $min_t\text{-minimum}_{\forall j \in X}(\text{min}(d_{ij}))+\text{minimum}_{\forall j \in X}(\text{min}(d_{ij})) \leq min_t\text{-minimum}_{\forall j \in X}(\text{min}(d_{ij}))+\text{minimum}_{\forall j \in X}(d_{ij}) \leq T$. As a result, $min_t \leq min_{tp} \leq T$. Similarly, the maximum time T required to execute the SP is calculated such that $T \leq max_t\text{-maximum}_{\forall j \in X}(\text{max}(d_{ij}))+\text{maximum}_{\forall j \in X}(d_{ij})$. Since $\text{maximum}_{\forall j \in X}(d_{ij}) \leq \text{maximum}_{\forall j \in X}(\text{max}(d_{ij}))$, then $T \leq max_t\text{-maximum}_{\forall j \in X}(\text{max}(d_{ij}))+\text{maximum}_{\forall j \in X}(d_{ij}) \leq max_t\text{-maximum}_{\forall j \in X}(\text{max}(d_{ij}))+\text{maximum}_{\forall j \in X}(\text{max}(d_{ij}))$. Therefore, $T \leq max_{tp} \leq max_t$. As a result, in this final case, in the PEs, the time T required for executing an SP is T such that $min_t \leq min_{tp} \leq T \leq max_{tp} \leq max_t$.

As a result, for all cases, in the PEs, the time T required for executing an SP is $min_{tp} \leq T \leq max_{tp}$ such that the time interval $[min_t, max_t]$ is associated with the corresponding transition in the S-SPEC and $min_t \leq min_{tp} \leq T \leq max_{tp} \leq max_t$. ■

Lemma 2. For any sequence of SPs in the S-SPEC executed during the time interval $[min_t, max_t]$, the corresponding SPs in the PEs are executed within the same or narrowed time interval.

Proof: The execution of sequence of n SPs in the S-SPEC is performed during the time interval $[min_t, max_t]$ such that $min_t = min_t(\text{SP}_1)+ min_t(\text{SP}_2)+ \ldots + min_t(\text{SP}_n)$ and $max_t = max_t(\text{SP}_1)+ max_t(\text{SP}_2)+ \ldots + max_t(\text{SP}_n)$. By Lemma 1, for any SP, $min_t(\text{SP})$ in the PE-SPEC is greater than or equal to the $min_t(\text{SP})$ in the S-SPEC and $max_t(\text{SP})$ in the PE-SPEC is less than or equal to the $max_t(\text{SP})$ in the S-

SPEC. Therefore, the execution of the sequence of n SPs in the PE-SPECs is performed within the same or narrowed time interval $[min_t, max_t]$. ■

Lemma 3. The time constraints assigned to the transitions of the PEs as a result of applying the extended synthesis method conform to the time constraints assigned to the transitions of the S-SPEC.
Proof: As a result of assigning time intervals to the transitions of the PEs using the extended synthesis method, the execution of any sequence of SPs in the PEs is performed during the same or narrowed time intervals given in the S-SPEC (Lemma 2). Therefore, the time constraints assigned to the transitions of the PEs as a result of applying the extended synthesis method conform to the time constraints assigned to the transitions of the S-SPEC. ■

6. Conclusions and Future Work

In this paper, a synthesis method for protocol specifications from service specifications is extended such that the timing constraints provided in the service specification are considered in the resulting protocol specifications. This extension makes the synthesis method applicable for real time applications. The extension uses the TFSM for modeling both the service and protocol specifications. In this paper, the assignment of the timing constraints to the service specification is discussed. In addition, it is shown how to map the timing constraints associated with the transitions of the service specification model to the transitions of the protocol specification models. The maximum and minimum delays of the channels between the protocol entities are considered in this paper when mapping the timing constraints. Finally, the introduced extension is proved to be correct in terms of the conformation of the timing constraints computed for the protocol specifications to the timing constraints provided in the service specification.

The basic synthesis method extended in this paper is limited to the service specifications that have sequential behavior (i.e., only one service primitive can be executed at once). In future, we plan to extend the basic synthesis method to handle possible concurrent occurrence of service primitives in the service specifications. In addition, we intend to study the affect of the concurrent behavior of the service specification on the assignment of the time constraints to the service and protocol specifications.

References

[1] R. Probert and K. Saleh, Synthesis of communication protocols: survey and assessment, *IEEE Transactions on Computers, Special Issue on Protocol Engineering*, 40(4), 1991, pp. 468-475.

[2] K. Saleh and R. Probert, Automatic synthesis of protocol specifications from service specifications, *Proceedings of the 10th IEEE International Phoenix Conference on Computers and Communications (IPCC-91)*, March 1991, pp. 615-621.

[3] G.V. Bochmann and R. Gotzhein, Deriving protocol specifications from service specifications, *Proceedings of SIGCOMM'86*, 1986, pp. 144-156.

[4] P.M. Chu and M.T. Liu, Synthesizing protocol specifications from service specification in the FSM model, *Proceedings of Computer Networking Symposium*, 1988, pp. 173-182.

[5] Z.P. Tao and M. Goossens, Synthesizing communication protocol converter: a model and method, *Proceedings of the 1992 ACM Annual Conference on Communications*, Kansas City, Missour, US, 1992, pp. 17-24

[6] R. Gotzhein and G.V. Bochmann, Deriving protocol specifications from service specifications including parameters, *ACM Transactions on Computer Systems*, 8(4), 1990, pp. 255-283.

[7] A. Khoumsi, New results for deriving protocol specifications from service specifications for real-time applications, *Proceedings of the Maghrebian Conference on Software Engineering and Arterial Intelligent (MCSEAI)*, Tunis, Tunisia, December 1998.

[8] M. Kapus-Kolar, Deriving protocol specifications from service specifications with heterogeneous timing requirements, *Proceedings of IEEE 3rd International Conference on Software Engineering for Real-Time Systems*, Cirenchester, UK, 1991, pp. 1093-1096.

[9] J. Park and R.E. Miller, Synthesizing protocol specifications from service specifications in timed extended finite state machines, *Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97)*, 1997, pp. 253.

[10] T. Higashino, K. Okano, H. Imajo, K. Taniguchi, Deriving protocol specifications from service specification in extended FSM models, Proceedings of the 13th International Conference on Distributed Computing Systems, 1993, pp. 141-148.

[11] H. Yamaguchi, K. Okano, T. Higashino, K. Taniguchi, Protocol synthesis from time Petri Net based service specifications, Proceedings of the 1997 International Conference on Parallel and Distributed Systems, 1997, pp. 236 – 243.

[12] H. Yamaguchi, K. El-Fakih, G. von Bochmann, and T. Higashino, Protocol synthesis and re-synthesis with optimal allocation of resources based on extended Petri nets, *Distributed Computing,* 16(1), 2003, pp. 21-35

[13] T.Y. Choi, Sequence method for protocol construction, *Proceedings of the 6th IFIP International Symposium on Protocol Specification, Testing, and Verification*, 1986, pp. 307-321.

[14] Y.X. Zhang, K. Takahashi, N. Shiratori, and S. Noguchi, An interactive protocol synthesis algorithm using a global state transition graph, *IEEE Transactions on Software Engineering*, SE-14(4), 1988, pp. 394-404.

[15] W.A. Barrett and J.D. Couch, *Compiler Construction: Theory and Practice, Chapter 3*, Science Research Associates, 1979.