A Systematic and Optimized Method for Designing Protocols for Real-Time Applications

A. Khoumsi, G.v. Bochmann
R. Dssouli, A. Ghedamsi

Publication # 900

Département d'informatique et de recherche opérationnelle

Université de Montréal

Avril 1994

A SYSTEMATIC AND OPTIMIZED METHOD FOR DESIGNING PROTOCOLS FOR REAL-TIME APPLICATIONS

A. Khoumsi , G.v. Bochmann , R. Dssouli, A. Ghedamsi

Université de Montréal
Faculté des arts et des sciences
Département d'informatique et
de recherche opérationnelle
C.P. 6128, Succursale A
Montréal, (Quebec)
H3C 3J7

Abstract. In [KBD93] and in this paper, service and protocol are specified by timed automata. In [KBD93], a method for deriving real-time protocol specifications from service specifications is proposed. In this paper, we improve and generalize this method. Improvement is made by minimizing the number of exchanged messages between protocol entities. In this case, temporal requirements on protocol are less strong than in [KBD93]. Generalization is made by considering an unreliable medium. An error-recovery capability is then necessary.

1. Introduction

A way for specifying real-time applications is to use timed automata, where executions of transitions are associated to temporal conditions. In this paper conditions represent temporal requirements only between consecutive transitions. For instance, we can specify that the delay between a data transmission and its reception must be less than t_{max}. More generally, a time between two consecutive events must be in an interval [t_{min}, t_{max}]. In [KBD93], we propose a method for generating timed automata specifying the protocol from a timed automaton specifying the desired service. In this paper, we firstly improve this method by proposing a way for reducing the number of synchronization messages exchanged between protocol entities. We show that the temporal requirements synthesized for protocol entities are less strong than those generated in [KBD93]. Secondly, we show that our method can be used even if the medium is unreliable, provided that few modules are added to protocol entities: one module per protocol entity.

The continuation of this paper is organized as follows. In section 2, we show how service and protocol for non-real-time applications are specified. In section 3, we introduce the basic principle for deriving protocol entities, and we improve the way this principle is used in [KBD93] by minimizing the number of exchanged messages. Afterwards, we present the rules for deriving protocol without real-time requirements. In section 4, we describe how temporal requirements are specified in the service and the protocol. In section 5, we explain the approach used for calculating temporal requirements for protocol entities from temporal requirements on the service. In section 6, the resolution is done for three cases, one static and two dynamic. We show that the obtained temporal requirements are less strong than those in [KBD93]. In section 7, we present the different steps used for deriving protocol specifications for real-time applications. In section 8, few examples illustrate our method. In section 9, we consider that the medium is unreliable. We show that the protocol entities synthesized for a reliable medium can be used for an unreliable medium. In this case, a protocol entity communicates with the medium via a module which makes the unreliability of the medium invisible by the protocol entity. And finally, we conclude.

2. Service and protocol specifications for non real-time applications

2.1. Service specification

The desired service is described by a finite automaton, noted SS, which specifies the sequences of service primitives (SP) we would like to observe at the different service access points (SAP). To each SAP corresponds one protocol entity (PE) and we will not distinguish a PE and the corresponding SAP. Transitions of SS are defined by three parameters (fig.1) which are :

- the service primitive E executed by the transition
- a number a identifying the entity or the SAP where the service primitive E is executed. This entity is noted PE_a
- a number p identifying the transition, which is then noted T_p=(E,a)

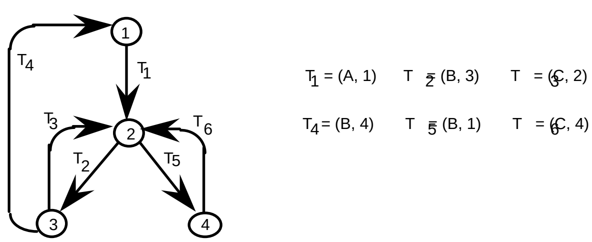


Figure 1. Service specification

A transition is then designated by T_p=(E,a) and means that the primitive E is executed in PE_a. As in [SP90, KBD93], for a state e of SS, out(e) et in(e) are respectively the sets of SAP

corresponding to the outgoing and ingoing transitions. Example : on figure 1 in(2) = {SAP_1,
SAP_2, SAP_4}, and out(2)={SAP_1,SAP_3}.

2.2. Protocol specification

A protocol entity PE_a is described by a finite automaton, noted PS_a (fig. 6), which has three types of transitions.

First type : execution of a service primitive
Second type : the sending of a message is defined by $s_i(p)$, and means "message parameterized by p is sent by PE_a to entity PE_i".
Third type : reception of a message is defined by $r_i(p)$, and means "message parameterized by p is received by PE_a from PE_i".

3. Deriving protocol entities for non real-time applications

3.1. Principle of derivation

Deriving protocol consists on generating as many finite automata as the number of protocol entities. Each of these automata is noted PS_i and specifies the protocol entity PE_i. For providing the desired service, the different PE_i will exchange synchronization messages through a reliable medium. The basic principle used for deriving protocol is rather simple : when in the service two consecutive primitives A and B are executed by two different entities PE_a and PE_b, then :

- after execution of A by PE_a, this one sends a message m to entity PE_b
- after reception of message m by PE_b, this one executes B

If after execution of the service primitive A by PE_a, there is a choice between k service primitives B_i executed by PE_{bi} (i=1 to k) (fig.2), the basic principle is then used in [KBD93] as follows. When PE_a executes transition T_p, it decides which transition among T_{pi} (i=1 to k) must be executed. It sends then the same message to all PE_{bi} ($\neq PE_a$). The message contains the following two parameters:

- the identifier p of the executed transition T_p,
- the identifier pj of the chosen transition T_{pj} to be executed.

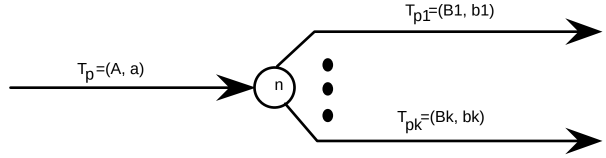


Figure 2. Choice between several actions.

All entities PE_{b1} to PE_{bk} receive the message sent by PE_a but only the chosen entity executes its transition. With this method, PE_a may possibly send an important number of messages to inform one entity that it can execute its transition, and all other entities that they must do nothing. For a state e of SS, the number of messages is equal to the cardinal of out(e), noted |out(e)|. Our improvement here is that PE_a must send only one message, to the selected entity to inform it that it can execute one of its transitions.

3.2. Rules for deriving protocol entities

3.2.1. Transformation of the service specification

The first step for deriving protocol is to transform the service specification SS into an equivalent specification TSS (T for Transformed). The latter must respect the following condition.

C1: From every state e of TSS, all executable outgoing transitions are executed by a same protocol entity PE_b (fig. 3), i.e. cardinal of out(e) is equal to one (|out(e)|=1).

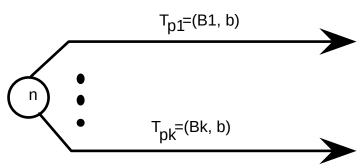
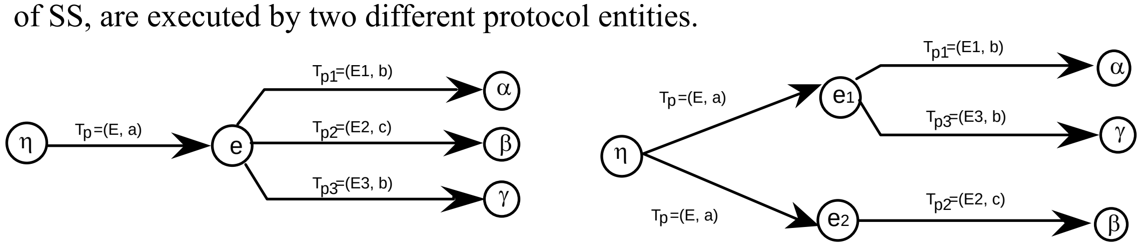


Figure 3. Outgoing transitions in a state of the transformed specification TSS.

The way for obtaining TSS from SS is the following. For every state e of SS, e is replaced by as many states e_i as the cardinal of out(e) (fig. 4) . Outgoing transitions from states e_i respect the condition *C1* and the following condition.

C2: Outgoing transitions of two different states e_i and e_j of TSS, generated from a same state of SS, are executed by two different protocol entities.



4.a. State e in SS **4.b.** Transformation of e in TSS

Figure 4. Example of transformation from SS to TSS

Remarks : - if TSS≠SS, then TSS is non deterministic.

- If for every state e of SS, |ou(e)|=1, then TSS=SS.

The transformation of the service specification of figure 1 gives the equivalent specification on figure 5.

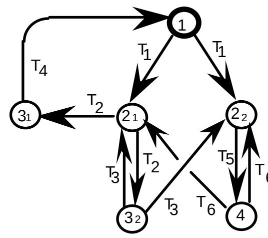


Figure 5. Transformation of specification in figure 1

3.2.2. Rules

From the sevice specification SS, the derivation procedure consists of five steps.

Step 1 : SS is transformed into TSS

Step 2 : From TSS, we generate GPS (global protocol specification) with the following rules :

For a transition T_p=(E,a) : $$n1 \xrightarrow{T_p=(E,a)} n2$$

Case a : if out(n2)={SAP_a}, the transition remains unchanged.

Case b : if out(n2)={SAP_b}≠{SAP_a}, the transition becomes:

$$n1 \xrightarrow{T_p=(E,a)} \bigcirc \xrightarrow{t_b^a(p)} n2$$

where $t_b^a(p)$ means "message parameterized by p is sent by PE_a and then received by PE_b ".

Step 3 : For each PE_i, we generate GPS_i from GPS by the following rules :

For a transition T_p=(E,a) : $$n1 \xrightarrow{T_p=(E,a)} n2$$

Case a : if a=i, the transition becomes : $$n1 \xrightarrow{E} n2$$

Case b : if a≠i the transition becomes: $$n1 \xrightarrow{\varepsilon} n2$$

where ε represents a spontaneous transition.

For a transition $t_b^a(p)$: $$n1 \xrightarrow{t_b^a(p)} n2$$

Case a : if a=i (then b≠i), the transition becomes : $$n1 \xrightarrow{s_b(p)} n2$$

Case b : if b=i (then a≠i), the transition becomes: $$n1 \xrightarrow{r_a(p)} n2$$

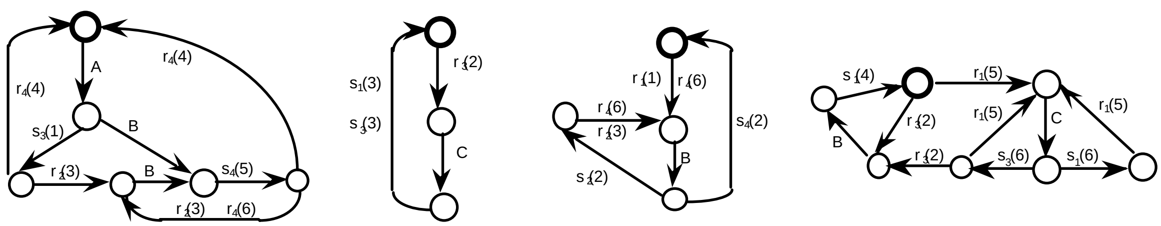
Case c : if a≠i and b≠i, the transition becomes: $$n1 \xrightarrow{\varepsilon} n2$$

where $s_b(p)$ means " message parameterized by p is sent to PE_b ",

and $r_a(p)$ means " message parameterized by p and coming from PE_a is received"

Step 4 : Transitions ε of the different GPS_i are considered spontaneous and are removed by projection for obtaining protocol specifications PS_i . An algorithm for removing ε is given in [BC79]. Intuitively, let Aε be an automaton containing transitions ε and specifying a system $\mathcal{A}$, and let A be the automaton obtained by removal of ε from Aε. If an external observer can detect all transitions but ε, then A is the specification of $\mathcal{A}$ as it is perceived by the observer.

Step 5 : The obtained PSi are minimized, and are transformed into deterministic automata if they are non deterministic.
For our example in figure 5, we obtain the specifications in figure 6:



6.a. PS_1 **6.b.** PS_2 **6.c.** PS_3 **6.d.** PS_4

Figure 6. Obtained protocol specifications .

We can prove that the unique obtained solution is *semantically and syntactically correct* . The semantics is correct means that the derived entities provide the service specified by SS. Their syntax is correct because they are deadlock-free and livelock-free, and no unspecified reception error is possible.

4. Service and protocol specifications for real-time applications

4.1. Service specification

On a service specification with time requirements (SST), each transition is defined by :
- the three parameters presented in the previous section,
- a set C_p of time intervals, where p is the number identifying the transition .

A transition is then defined by T_p=(E,a,p,C_p), and the execution of T_p means execution by entity PE_a of action E of the transition T_p. Let's consider for a state n of SST, its k ingoing transitions T_{pi}, and its m outgoing transitions T_{qj} (figure 7). The representation of figure 7 is used for defining the semantics of the sets C_{qj} of the outgoing transitions . Each C_{qj} contains as many time intervals as there are ingoing transitions on state n, i.e. it contains k intervals noted $T_{pi,qj}=[T^{mi}_{pi,qj}; T^{ma}_{pi,qj}]$ (i= 1 to k). The semantics of a $T_{pi,qj}$ is the following :

When state n is reached by an ingoing transition T_{pi}, then :

Condition 1 : if the transition T_{qj} is executed, it must be executed in the time interval $T_{pi,qj}$ after state n has been reached.

Condition 2 : besides, an outgoing transition among all the transitions T_{qj} (j = 1 to m) must inevitably be executed after state n is reached.

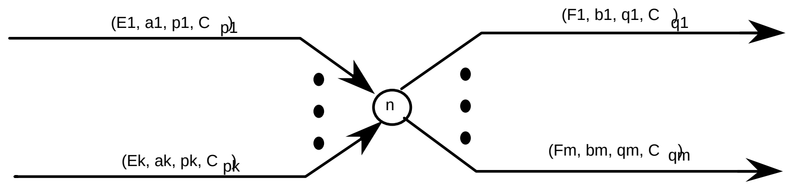


Figure 7. Ingoing and outgoing transitions on a state of SST

Example: let n be a state with one ingoing transition and two outgoing transitions (fig.8).

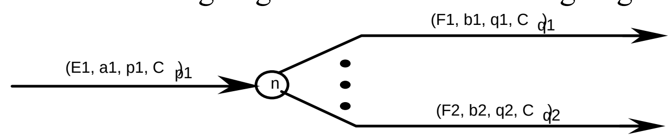


Figure 8. Example on the definition of the semantics of time intervals

Each of C_{q1} and C_{q2} contains one interval , with $C_{q1}=\{T_{p1,q1}\}$ and $C_{q2}=\{T_{p1,q2}\}$. For example $T_{p1,q1}=[1,3]$ and $T_{p1,q2}=[2,5]$. In this case, if T_{q1} (resp. T_{q2}) is executed, it must be executed in the interval [1,3] (resp. [2,5]) after execution of T_{p1} (condition 1). Besides, if neither T_{q1} nor T_{q2} are executed in a time equal to 3 after execution of T_{p1}, then T_{q2} must inevitably be executed in the interval [3,5] after execution of T_{p1} (condition 2). With this condition we have no deadlocks due to time constraints.

From this semantics, we deduce that if *state n is the initial state* then the different intervals of each C_{qj} are equal. In other words, for each j=1 to m, we have $T_{p1,qj}=T_{p2,qj}=...=T_{pk,qj}$.

Remark: T_p is a transition identified by p, while $T_{p,q}$ is the time interval containing the delay between transitions T_p et T_q.

4.2. Protocol specification for real-time applications

There are three types of transitions in a protocol specification.

First type : execution of a service primitive is defined by (E, D_p) where :
- E is the name of the service primitive,
- D_p is a set of intervals whose semantics is given in section 6.

Second type : sending a message to another protocol entity is defined by $s_i(p)\{S_{p,b}\}$, where $S_{p,b}$ is an interval, contrary to C_p and D_p which are sets of intervals, (fig. 13).

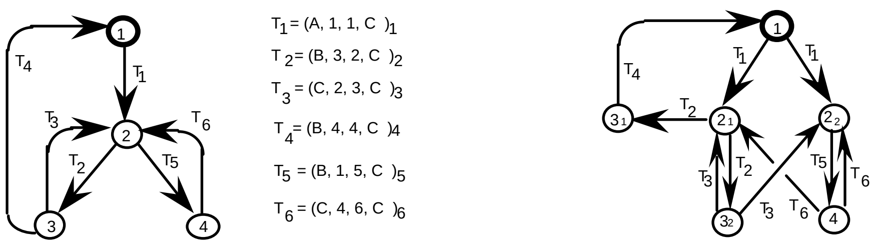
Third type : receiving a message is defined by $r_i(p)$. There is no time requirement in this type of transition (fig. 13). Time requirements in types one and two are sufficient for respecting time requirements in the service .

5. Approach of the problem for calculating time requirements (PCTR)

5.1. Transforming SST into TSST

Before calculating temporal requirements for protocol entities, the transformation presented in section 3.2 must be applied to the timed service specification SST for obtaining the specification TSST. Therefore, outgoing transitions of a same state in TSST are executed by a same protocol entity.

Example of figure 1 is reconsidered for a real-time application (fig. 9.a). After transformation, we obtain the non deterministic specification of figure 9.b.



9.a. Specification SST **9.b.** Specification TSST

Figure 9. Example of transformed real-time specification

5.2. Approach of the problem

For calculating time constraints on protocol entities, we must consider, at once on a state n of TSST, one of its ingoing transitions and all its outgoing transitions (fig.10). In a first time, we consider the case where out(n)≠{SAP_a}. In other words, the protocol entity PE_a (executing the ingoing transition) is different than PE_b which executes the outgoing transitions.

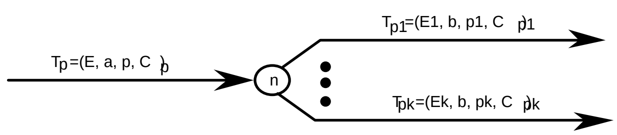
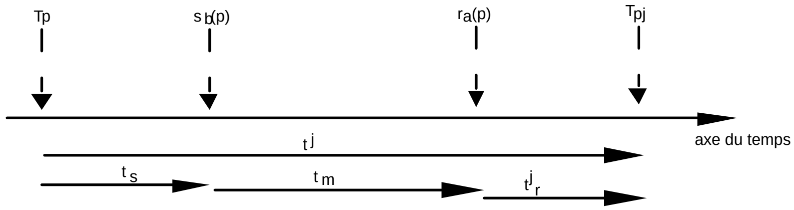
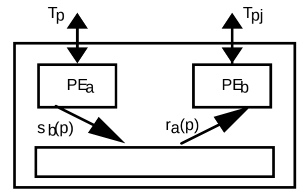


Figure 10. Outgoing transitions on a state of TSST

Let then T_p be a transition in an entity PE_a followed by several transitions T_{pj}, j=1 to k, in PE_b. After T_p, PE_a must send a message to PE_b (by $s_b(p)$), and when PE_b receives the message (by $r_a(p)$), it executes one of the k transitions T_{pj} (j=1 to k). The sequencing of events between T_p and T_{pj} is represented in function of the time on figure 11.a.



11.a. Representation in function of the time



11.b. Representation by entity

Figure 11. Representation of events between T_p and T_{pj}

The temporal requirements in the service impose that the time t^j between executions of T_p and T_{pj} belongs to $T_{p,pj}=[T^{mi}_{p,pj}; T^{ma}_{p,pj}]$. We suppose that we have a model of the reliable medium, i.e. the transit delay t_m , in the medium, of a message sent by PE_a and received by PE_b, belongs to an interval $M_{a,b}=[M^{mi}_{a,b}, M^{ma}_{a,b}]$ which depends on PE_a and PE_b.

The aim of temporal requirements derivation on protocol entities is the the following one.

From requirements $t_m \in M_{a,b}$ *and* $t^j \in T_{p,pj}$ *(for j= 1 à k), we must derive constraints on* t_s *and* t^j_r *(j=1 to k) which ensure that temporal requirements* $t^j \in T_{p,pj}$ *on the service will be respected . These derived constraints are written in the form* $t_s \in S_{p,b}=[S^{mi}_{p,b} ,S^{ma}_{p,b}]$, *and* $t^j_r \in R_{p,pj} =[R^{mi}_{p,pj} , R^{ma}_{p,pj}]$ *for j=1 à k .*

Requirements on t_s and t^j_r are temporal requirements on the protocol. In fact, t_s is the delay between T_p and $s_b(p)$ which are executed in PE_a, and t^j_r is the delay between $r_a(p)$ and T_{pj} which are executed in PE_b (fig. 11.b).

Remark : If $PE_a=PE_b$, no message is sent. In this case we take ts=tm=0. So the derivation is trivial.: $t^j=t^j_r$, then $S_{p,b}=[0;0]$, $M_{a.b}=[0;0]$ and $R_{p,pj}=T_{p,pj}$.

The following notations also will be used :

- $V^{mi}_{p,b}$ et $V^{ma}_{p,b}$ are parameters belonging to [0,1]. They are defined for a transition T_p (executed by an entity PE_a) and a protocol entity PE_b ($\neq PE_a$) which executes transitions consecutive to T_p. They are used to choose one solution among an infinite number of solutions. If we obtain, as we will see formerly, for $S_{p,b}=[S^{mi}_{p,b}, S^{ma}_{p,b}]$ the constraint $S^{ma}_{p,b} \in [\rho,\eta]$, we choose $S^{ma}_{p,b} = \rho+V^{ma}_{p,b}*(\eta-\rho)$. In the same manner, if we obtain $S^{mi}_{p,b} \in [\chi,\xi]$, we choose $S^{mi}_{p,b} = \chi + V^{mi}_{p,b}*(\xi-\chi)$.

- Addition and subtraction of two intervals [a, b] and [c, d] are defined by
  [a, b] + [c, d] = [a + c, b + d] , and [a, b] - [c, d] = [a - c, b - d].

If we summarize, the *entries of PCTR* for protocol entities are :

- $T_{p,q}$ and $M_{a,b}$ for every pair of consecutive transitions T_p et T_q , respectively executed in PE_a and PE_b,
- $V^{mi}_{p,b}$ and $V^{ma}_{p,b}$ for every pair (T_p, PE_b) of transition T_p and entity PE_b executing

transitions consecutively to T_p. They are used to choose a particular solution among an infinite number of solutions.

Solutions of PCTR are :

- $S_{p,b}$ for every transition T_p executed by an entity PE_a and followed by transitions executed in $PE_b \neq PE_a$. If $PE_b = PE_a$, we can take $S_{p,b} = [0;0]$.
- $R_{p,q}$ for every pair of consecutive transitions T_p and T_q .

We show in the next section 5.3 that there exist conditions on entries $T_{p,q}$ and $M_{a,b}$ of PCTR for the existence of solutions.

5.3. Condition for existence of solutions

We consider then, for a state n of TSST, one of its ingoing transition T_p (executed in PE_a), and all its outgoing transitions T_{pj} , j=1 to k, (executed in PE_b) (fig. 10). From figure 11.a, we can write :

$$\text{for } j=1 \text{ to } k: \quad t^j \in T_{p,pj} \text{ implies } t_s + t_m + t^j_r \in T_{p,pj} \tag{1}$$

As $t_m \in M_{a,b} = [M^{mi}_{a,b}, M^{ma}_{a,b}]$, then condition (1) implies :

$$\text{for } j=1 \text{ to } k: \quad t_s + M^{mi}_{a,b} + t^j_r \geq T^{mi}_{p,pj} \tag{2}$$

$$t_s + M^{ma}_{a,b} + t^j_r \leq T^{ma}_{p,pj} \tag{3}$$

Formulae (2) et (3) imply

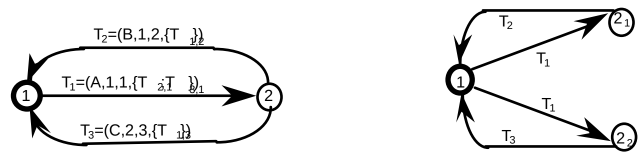
$$\text{for } j=1 \text{ to } k: \quad T^{ma}_{p,pj} - M^{ma}_{a,b} \geq \sup(T^{mi}_{p,pj} - M^{mi}_{a,b}; 0) \tag{4}$$

Condition (4) is for a state of TSST and one of its ingoing transitions. Therefore, for every state and one of its ingoing transitions T_p, resolution of PCTR consists in :

- checking if condition (4) is respected
- if the checking is positive then :
  - * interval $S_{p,b}$ is calculated (constraint on t_s),
  - * intervals $R_{p,pj}$, j=1 to k, are calculated (constraints on t^j_r , j=1 to k).

5.4. Comparison with [KBD93] approach (old approach)

In [KBD93], the resolution of PCTR is done from the specification SST. In our improved approach, the resolution is done from the transformed specification TSST. Therefore, condition (4) with the old approach is more restricting than here. In fact, a condition of existence from SST can be constituted by several conditions of existence from TSST, and it is respected if all those conditions are respected. An example is given on figure 12.



12.a. SST **12.b.** TSST

Figure 12. Example for comparing conditions for existence of solutions

From SST, we have three conditions of existence (i, j, and k), which are the following.

- for state 1 and ingoing transition T_2 : $T^{ma}{}_{2,1} \geq T^{mi}{}_{2,1}$ (i)
- for state 1 and ingoing transition T_3 : $T^{ma}{}_{3,1}$ - $M^{ma}{}_{2,1} \geq \sup(T^{mi}{}_{3,1}$- $M^{mi}{}_{2,1}$; 0) (j)
- for state 2 and ingoing transition T_1 : $T^{ma}{}_{1,2} \geq T^{mi}{}_{1,2}$ (k1)

  : $T^{ma}{}_{1,3}$ - $M^{ma}{}_{1,2} \geq \sup(T^{mi}{}_{1,3}$- $M^{mi}{}_{1,2}$; 0) (k2)

The third condition (k) is constituted by (k1) and (k2). If for instance only (k1) is respected, then the condition (k) is not respected, and we consider that the temporal requirements cannot be respected after execution of transition T_1. The last assumption is too restricting, because in reality only temporal requirements between executions of T_1 and T_3 cannot be respected (because k2 is not respected).

With the improved approach,the restriction does not exist: from TSST we have four conditions for existence of solutions, because (k1) and (k2) are considered to be two independent conditions. In fact (fig.12.b) (k1) is for temporal requirements between T_1 and T_2,

(k2) is for temporal requirements between T_1 and T_3.

The transformation which modifies the timed service specification SST into TSST has then a second advantage. Besides minimizing the number of messages, sometimes we can have solutions for PCTR from TSST when there are not from SST.

6. Resolution of PCTR

For resolving PCTR, we consider the three following cases :

Static case : messages transmitted by entities contain no temporal information ,
First dynamic case : the PE put a temporal information in messages they send,
Second dynamic case : besides the temporal information put by the PE, the medium adds a second temporal information in the message. This information is an estimation of the transit delay of the message in the medium. In this third case, treated in detail in section 6.3, the receiving entity can have a good temporal information without using a global clock.

6.1. Static case

This case is static because the intervals $S_{p,b}$ and $R_{p,pj}$ are constant. When an entity PE_a executes a transition T_p and decides to send a message to entity PE_b, the time t_s, between execution of T_p and transmission of the message, belongs to a constant interval $S_{p,b}$. When PE_b receives the message from PE_a, it can execute a transition T_{pj}, among the k possible transitions (j=1 to k), in a time t^j_r belonging to a constant interval $R_{p,pj}$.

If $S_{p,b}$ and $R_{p,pj}$ are such that condition (1) is respected for every $t_s \in S_{p,b}$ and $t^j_r \in R_{p,pj}$ and $t_m \in M_{a,b}$, then it is equivalent to have :

$$\text{for } j=1 \text{ to } k: \quad S_{p,b} + M_{a,b} + R_{p,pj} \subseteq T_{p,pj} \tag{5}$$

that is to say :

$$\text{for } j=1 \text{ to } k: \quad S^{mi}_{p,b} + M^{mi}_{a,b} + R^{mi}_{p,pj} \geq T^{mi}_{p,pj} \tag{6}$$

$$S^{ma}_{p,b} + M^{ma}_{a,b} + R^{ma}_{p,pj} \leq T^{ma}_{p,pj} \tag{7}$$

Resolution :

Condition (7) implies :

$$S^{ma}_{p,b} \in [0;\ \min_{j=1 \text{ à } k} (T^{ma}_{p,pj} - M^{ma}_{a,b})] \tag{8}$$

then (6) and (7) imply :

$$S^{mi}_{p,b} \in [\sup(U, 0);\ S^{ma}_{p,b}] \tag{9}$$

$$\text{with } U = \max_{j=1 \text{ à } k} (S^{ma}_{p,b} + (M^{ma}_{a,b} - M^{mi}_{a,b}) - (T^{ma}_{p,pj} - T^{mi}_{p,pj})) \tag{10}$$

By using parameters $V^{ma}_{p,b}$ and $V^{mi}_{p,b}$, we choose a particular solution for $S^{ma}_{p,b}$ and $S^{mi}_{p,b}$ which respects (8) and (9). We have then :

$$S^{ma}_{p,b} = V^{ma}_{p} * \min_{j=1 \text{ à } k} (T^{ma}_{p,pj} - M^{ma}_{a,b}) \tag{11}$$

$$S^{mi}_{p,b} = \sup(U, 0) + (S^{ma}_{p,b} - \sup(U, 0)) * V^{mi}_{p,b} \tag{12}$$

We choose afterwards the less restrictive solutions on $R_{p,pj}=[R^{mi}_{p,pj}; R^{ma}_{p,pj}]$ respecting (6) and (7). We have then :

$$\text{for } j=1 \text{ to } k: \quad R^{ma}_{p,pj} = T^{ma}_{p,pj} - M^{ma}_{a,b} - S^{ma}_{p,b} \tag{13}$$

$$R^{mi}_{p,pj} = \sup(T^{mi}_{p,pj} - M^{mi}_{a,b} - S^{mi}_{p,b};\ 0) \tag{14}$$

We can easily check that the obtained service is included in the desired service (safety).

It is better to choose $V^{ma}_{p,b}$ as small as possible and $V^{mi}_{p,b}$ as big as possible. This implies to have $S^{ma}_{p,b}$ and $S^{mi}_{p,b}$ as small and close as possible. $R^{ma}_{p,pj}$ and $R^{mi}_{p,pj}$ will be then the less constrained as possible, and the receiving entities will have as much time as possible to provide the service.

6.2. First dynamic case

This case is dynamic because the receiving PE_b calculates dynamically the interval $R_{p,pj}$, when it receives the message from PE_a. In fact, after execution of T_p, PE_a sends to PE_b a message with information t_s. And PE_b calculates $R_{p,pj}$ in function of t_s.

Resolution:

formula (3) implies $t_s + M^{ma}_{a,b} \leq T^{ma}_{p,pj}$. If $S_{p,b}=[S^{mi}_{p,b};S^{ma}_{p,b}]$ is an interval always containing t_s then we have the condition (8) as in the static case:

$$S^{ma}_{p,b} \in [0;\ \min_{j=1\ \grave{a}\ k} (T^{ma}_{p,pj} - M^{ma}_{a,b})] \tag{8}$$

And $S^{mi}_{p,b}$ is less constrained than in the static case :

$$S^{mi}_{p,b} \in [0 ; S^{ma}_{p,b}] \tag{15}$$

As in the static case , a particular solution is chosen by using parameters $V^{ma}_{p,b}$ and $V^{mi}_{p,b}$:

$$S^{ma}_{p,b} = V^{ma}_{p,b} * \min_{j=1\ \grave{a}\ k} (T^{ma}_{p,pj} - M^{ma}_{a,b}) \tag{11}$$

$$S^{mi}_{p,b} = S^{ma}_{p,b} * V^{mi}_{p,b} \tag{16}$$

If t_s , which belongs to $[S^{mi}_{p,b}; S^{ma}_{p,b}]$, is the delay when the message is sent after execution of T_p, the receiving entity knows it and can choose :

$$\text{for } j=1 \text{ to } k : \quad R^{ma}_{p,pj} (t_s) = T^{ma}_{p,pj} - M^{ma}_{a,b} - t_s \tag{17}$$

$$R^{mi}_{p,pj} (t_s) = \sup(T^{mi}_{p,pj} - M^{mi}_{a,b} - t_s;\ 0) \tag{18}$$

We can easily check that the provided service is included in the desired service. With the information t_s, the receiving entity PE_b will use the time allocated to it to provide the service more efficiently then in the static case. In fact, time interval $R_{p,pj}(t_s)$ ((17) and (18)) is less restricting than interval $R_{p,pj}$ ((13) and (14)), because $R_{p,pj}$ is strictly included in $R_{p,pj}(t_s)$. Intuitively, in dynamic case the receiving entity PE_b has a more accurate information about when T_p has been executed by PE_a. In the static case, it has to suppose the worst cases for the time t_s . Therefore, sometimes in static case it has to "hurry up", when in dynamic case it has not to.

6.3. Second dynamic case

In this case, PE_b receives the message with informations t_s and t_m, and it calculates dynamically the interval $R_{p,pj}$ in function of these two informations.

Resolution :

$S_{p,b} = [S^{mi}_{p,b} ; S^{ma}_{p,b}]$ is resolved as in section 6.2 ((11) and (16)). $R^{ma}_{p,pj}$ and $R^{mi}_{p,pj}$ are calculted dynamically by PE_b with the following formulae :

$$\text{for } j=1 \text{ to } k : R^{ma}_{p,pj} (t_s, t_m) = T^{ma}_{p,pj} - t_s - t_m \qquad (19)$$

$$R^{mi}_{p,pj} (t_s, t_m) = \sup(T^{mi}_{p,pj} - t_s - t_m;\ 0) \qquad (20)$$

We can check that the desired service is respected (safety) by the protocol. With information t_m, the receiving entity PE_b uses more efficiently the time allocated to it to provide the service. In fact, $R_{p,pj}(t_s)$ ((17) and (18)) is strictly included in $R_{p,pj}(t_s, t_m)$ ((19) and (20)).

6.4. Comparison with [KBD93] approach

Temporal requirements on protocol obtained with [KBD93] approach are more restricting than those derived with our improved approach. In fact, intervals containing $S^{ma}_{p,b}$ and $S^{mi}_{p,b}$ ((8), (9) and (15)) are bigger than or equal to those obtained in [KBD93]. With our approach, $S_{p,c}$ and $R_{p,pj}$ are independent when T_{pj} is executed by $PE_b \neq PE_c$.

If we recapitulate, advantages of the approach here are :
- the number of exchanged messages is minimized,
- conditions for existence of solutions are less strong,
- derived temporal requirements are less restricting.

6.5. Transit delay in the medium

In the second dynamic case, time t_m is not an accurate value. It is an estimation of the transit delay in the medium. In fact, if the message goes through many nodes before reaching its destination, t_m comprises estimations of :

* transmission and propagation delays between the different adjacent nodes,
* the time passed in the nodes (processing and especially waiting in queues) .

For these reasons, positive parameters α and β can be added in formulae (19) and (20) which become :

$$\text{for } j=1 \text{ to } k : R^{ma}_{p,pj} (t_s, t_m) = T^{ma}_{p,pj} - t_m - t_s - \beta \qquad (21)$$

$$R^{mi}_{p,pj} (t_s, t_m) = \sup(T^{mi}_{p,pj} - t_m - t_s + \alpha;\ 0) \qquad (22)$$

This is equivalent to estimate the transit delay in the interval $[t_m - \alpha ; t_m + \beta]$.

7. Deriving protocol for real-time applications

The derivation procedure consists of six steps.

Step 1 : The service specification SST is transformed into the equivalent TSST

Step 2 : From the specification TSST, we generate SSTT defined below.

- In the static case, a receiving entity PE_b must know the constant interval $R_{p,pj}$ ((13) and (14)) , which is the time interval allocated to it, since reception of the message, for executing T_{pj}.

- In the first dynamic case, a receiving entity PE_b must know the constant interval $X_{p,pj} = T_{p,pj} - M_{a,b}$ and the parameter t_s contained in the message. PE_b can therefore calculate dynamically $R_{p,pj}$ (by formulae (17) and (18)).

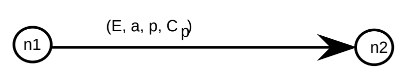
- in the second dynamic case, PE_b must know the constant interval $T_{p,pj}$, and parameters t_s and t_m received in the message. It can therefore calculate dynamically $R_{p,pj}$ (by formulae (19) and (20)).

We deduce from this that SSTT is obtained from TSST by :

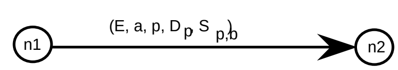
* associating time intervals $S_{p,b}$ to transitions T_p followed by transitions executed in $PE_b \neq PE_a$. Here PE_a is the entity which executes T_p.
* replacing time intervals $T_{p,pj}$ by intervals:
    - $R_{p,pj}$ in the static case
    - $X_{p,pj}$ in the first dynamic case

The substitution is not done in the second dynamic case.

For obtaining SSTT, every transition T_p of TSST :



is then replaced by the transition :



Where out(n2) ={SAP_b} and D_p is the set of intervals :

* $R_{pi,p}$ in the static case
* $X_{pi,p}$ in the first dynamic case
* $T_{pi,p}$ in the second dynamic case

where pi are identifiers of ingoing transitions of state n1.

Remark : Intervals in D_p have not the same semantics in the three cases. In the static case, $R_{pi,p}$ are constant temporal requirements, while in the two other cases, $X_{pi,p}$ and $T_{pi,p}$ are constant intervals used for dynamic calculation of the time requirements on T_p when it succeeds to transition T_{pi}.

The complexity of the algorithm for generating SSTT is in $O(n*e*s)$ with :

- n : number of states of TSST,
- e : maximum number of ingoing transitions by state in TSST,
- s : maximum number of outgoing transitions by state in TSST.

Step 3 : For each PE_i we generate SST_i from SSTT. The finite automaton SST_i is obtained by replacing every transition $(E,a,p,D_p,S_{p,b})$ by : - (ε,a,p) if $a \neq i$,
- $(E,p,D_p,S_{p,b})$ if $a=i$

Step 4 : A finite automaton $SPST_i$, is derived from each SST_i by using the following rules :

- For a transition $(E,p,D_p,S_{p,b})$ of SST_i : $n1 \xrightarrow{(E,\, p,\, D_p,\, S_{p,b})} n2$

Case a : if $out(n2) = \{SAP_i\}$ the transition becomes : $n1 \xrightarrow{(E,\, D_p)} n2$

In this case, interval $S_{p,b}$ is not defined because b=i.

Case b : if $out(n2) \neq \{SAP_i\}$ we obtain :

* in the static case : $n1 \xrightarrow{(E,\, D_p)} \bigcirc \xrightarrow{s_b(p)\{S_{p,b}\}} n2$

* in the two dynamic cases : $n1 \xrightarrow{(E,\, D_p)} \bigcirc \xrightarrow{s_b(p,t_s)\{S_{p,b}\}} n2$

$s_b(K)$ means "transmission to entity PE_b of message parameterized by K" . $\{S_{p,b}\}$ specifies that $s_b(K)$ must be executed in a time belonging to interval $S_{p,b}$ after the preceding action.

-For a transition (ε,a,p): $n1 \xrightarrow{(\varepsilon,a,p)} n2$

Case c : if $out(n2) =\{SAP_i\}$, the transition becomes :

* static case : $n1 \xrightarrow{r_a(p)} n2$

* first dynamic case : $n1 \xrightarrow{r_a(p,\, t_s)} n2$

* second dynamic case : $n1 \xrightarrow{r_a(p,\, t_s,\, t_m)} n2$

Case d : if $out(n2)=\{SAP_j\}\neq\{SAP_i\}$, the transition becomes : $n1 \xrightarrow{\varepsilon} n2$

$r_a(K)$ means "reception from PE_a of a message parameterized by K" .

Step 5: The transitions ε are considered spontaneous and are removed by projection (see also section 3.1.2). We obtain then timed protocol specifications for each PE_i.

Step 6 : The obtained specifications are minimized, and transformed into deterministic automata PST_i if they are non deterministic.

8. Examples

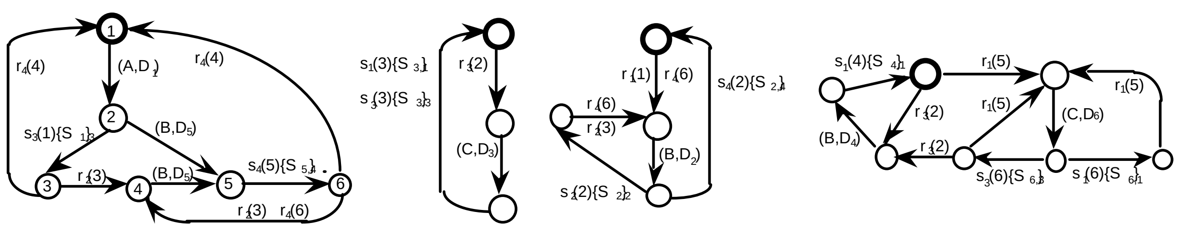
8.1. Example 1

We consider example of figure 9. This example is also in [KBD93] with the old approach. We have $C_1 = \{T_{4,1}\}$, $C_2 = \{T_{1,2} ; T_{3,2} ; T_{6,2}\}$, $C_3 = \{T_{2,3}\}$, $C_4 = \{T_{2,4}\}$, $C_5 = \{T_{1,5} ; T_{3,5} ; T_{6,5}\}$, $C_6 =\{T_{5,6}\}$.

For instance $T_{4,1}$=[3, 6], $T_{1,2}$ =[5,10], $T_{3,2}$=[4,8], $T_{6,2}$=[4,10] , $T_{2,3}$=[3,8] , $T_{2,4}$=[4,9] , $T_{1,5}$=[1,3] , $T_{3,5}$=[4,8] , $T_{6,5}$=[3,8] , $T_{5,6}$=[4,10].

We choose the medium $M_{u,v}$=[2,4] for every (u,v), and at last $V^{ma}_{p,a}=V^{mi}_{p,a}=0.5$ for $(p,a) \in \{(1,3);(2,2);(2,4);(3,1);(3,3);(4,1);(5,4);(6,1);(6,3)\}$.

The derived protocol specifications are represented below (figure 13).



13.a. PST_1 **13.b.** PST_2 **13.c.** PST_3 **13.d.** PST_4

Figure 13. First example of protocol specifications with time requirements

With $D_1=\{R_{4,1}\}$, $D_2=\{R_{1,2};\ R_{3,2};\ R_{6,2}\}$, $D_3=\{R_{2,3}\}$, $D_4=\{R_{2,4}\}$, $D_5=\{R_{1,5};\ R_{3,5};\ R_{6,5}\}$, $D_6=\{R_{5,6}\}$.

Remark: Step 5, which consists on removing non determinism, generates redundant temporal requirements. For example on figure 13.a, $D_5=\{R_{1,5}; R_{3,5}; R_{6,5}\}$. But for transition (B,D5) from state 2, only $R_{1,5}$ is necessary. And for transition (B,D_5) from state 4, only $R_{3,5}$ and $R_{6,5}$ are necessary.

$R_{1,5}$ contains the delay between executions of (A,D_1) and (B,D_5),

$R_{3,5}$ contains the delay between executions of $r_2(3)$ and (B,D_5),

$R_{6,5}$ contains the delay between executions of $r_4(6)$ and (B,D_5),

From formulae (11), (12), (13) and (14), we calculate :

$S_{1,3}$ = [1.5; 3]	$R_{1,2}$ = [1.5; 3]	$S_{4,1}$ = [0.5; 1]	$R_{4,1}$ = [0.5; 1]
	$R_{1,5}$ = [1; 3]	$S_{5,4}$ = [1.5; 3]	$R_{5,6}$ = [0.5; 3]

$S_{2,2} = [1; 2]$ $R_{2,3} = [0; 2]$ $S_{6,1} = [1; 2]$ $R_{6,5} = [0 ; 2]$

$S_{2,4} = [1.25; 2.5]$ $R_{2,4} = [0.75; 2.5]$ $S_{6,3} = [1.5; 2]$ $R_{6,2} = [0.5; 3]$

$S_{3,1} = [1; 2]$ $R_{3,5} = [1; 2]$

$S_{3,3} = [1; 2]$ $R_{3,2} = [1; 2]$

8.2. Example 2

This example also is in [KBD93] with the old approach. Two protocol entities PE_1 and PE_2 must communicate in a connected mode. To reduce calculations, we do the following hypotheses :

- connection and disconnection are done by PE_1 ,
- the provider of service in PE_2 cannot refuse a connection ,
- data transfer is done from PE_1 to PE_2 ,
- PE_1 sends a new data only if the preceeding has been received by PE_2 .

The executed events are noted TC.rqt, TC.ind, TC.rsp, TC.cnf, TD.rqt, TD.ind, TDt.rqt and TDt.ind . TC, TD and TDt are respectively abbreviations of T-connect, T-disconnect and T-data. And rqt, ind, rsp and cnf are respectively abbreviations of request, indication, response and confirm. A formal representation of service with time requirements is represented on figure 14, it is inspired by the protocol of the transport layer classe 0 ([Ta90], that is why primitives have names beginning by letter T). We have T_1=(TC.rqt,1,1,C_1), T_2=(TC.ind,1,2,C_2), T_3=(TC.ind,2,3,C_3), T_4=(TD.rqt,2,4,C_4), T_5=(TD.ind,1,5,C_5), T_6=(TC.rsp,2,6,C_6), T_7=(TC.cnf,1,7,C_7), T_8=(TDt.rqt,1,8,C_8), T_9=(TDt.ind,2,9,C_9), T_{10}=(TD.rqt,1,10,C_{10}), T_{11}=(TD.rqt,1,11,C_{11}), T_{12}=(TD.ind,2,12,C_{12}). Où $C_1=\{T_{2,1}, T_{5,1}, T_{12,1}\}$, $C_2=\{T_{1,2}\}$, $C_3=\{T_{1,3}\}$, $C_4=\{T_{3,4}\}$, $C_5=\{T_{4,5}\}$, $C_6=\{T_{3,6}\}$, $C_7=\{T_{6,7}\}$, $C_8=\{T_{7,8}, T_{9,8}\}$, $C_9=\{T_{8,9}\}$, $C_{10}=\{T_{7,10}, T_{9,10}\}$, $C_{11}=\{T_{8,11}\}$, $C_{12}=\{T_{10,12}, T_{11,12}\}$.

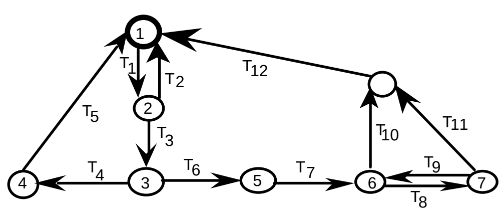


Figure 14. Formal specification of the desired service with two communicating entities

On figure 14:

- T_1 to T_7 correspond to connection set-up phase. If the connection is accepted, state 6 is reached.
- T_8 to T_9 represent data transfer phase,
- T_{10} to T_{12} specify the disconnection phase.

Let's take for instance $T_{2,1}=T_{5,1}=T_{12,1}=[3,6]$, $T_{1,2}=[1,2]$, $T_{1,3}=[3,7]$, $T_{3,4}=[1,2]$, $T_{4,5}=[2,5]$, $T_{3,6}=[2,3]$, $T_{6,7}=[4,7]$, $T_{7,8}=[1,3]$, $T_{9,8}=[2,6]$, $T_{8,9}=[3,6]$, $T_{9,10}=[2,5]$, $T_{8,11}=[0,2]$, $T_{10,12}=T_{11,12}=[3,6]$. Let's also take the medium Mu,v=[2,4] for every (u,v), and finally parameters $V^{ma}_{p,b}=V^{mi}_{p,b}=0.5$ for every p=1, 4, 6, and 8 to 12.

The derived protocol specifications with time requirements are represented on figures 15 and 16, with $D_1=\{R_{2,1}, R_{5,1}, R_{12,1}\}$, $D_2=\{R_{1,2}\}$, $D_3=\{R_{1,3}\}$, $D_4=\{R_{3,4}\}$, $D_5=\{R_{4,5}\}$, $D_6=\{R_{3,6}\}$, $D_7=\{R_{6,7}\}$, $D_8=\{R_{7,8}, R_{9,8}\}$, $D_9=\{R_{8,9}\}$, $D_{10}=\{R_{7,10}, R_{9,10}\}$, $D_{11}=\{R_{8,11}\}$, $D_{12}=\{R_{10,12}, R_{11,12}\}$.

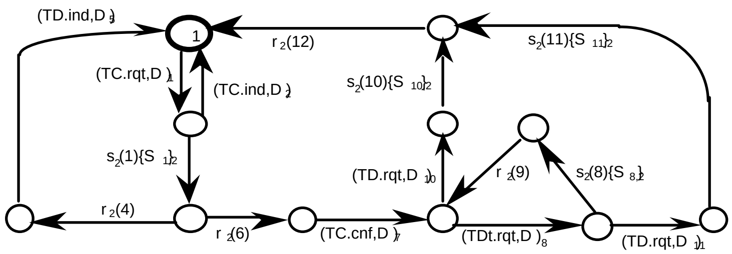


Figure 15. Protocol specification for the communicating entity PE_1

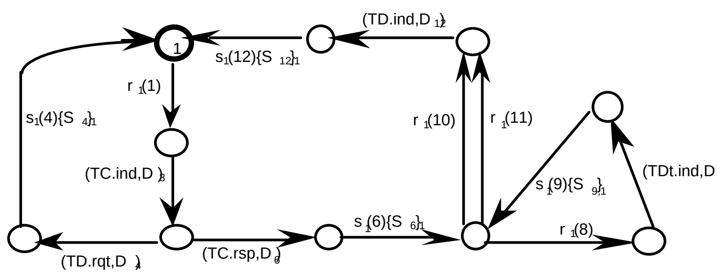


Figure 16. Protocol specification for the communicating entity PE_2

From formulae (11), (12), (13) and (14) we calculate:

$R_{1,2} = [1; 2]$

$S_{1,2} = [0.75; 1.5]$ $\quad R_{1,3} = [0.25; 1.5]$

$R_{2,1} = [3; 6]$ $\quad R_{3,4} = [1; 2]$ $\quad R_{3,6} = [2; 3]$

$S_{4,1} = [0.25; 0.5]$ $\quad R_{4,5} = [0; 0.5]$

$R_{5,1} = [3; 6]$

$S_{6,1}= [1; 1.5]$ $\quad R_{6,7} = [1; 1.5]$

$R_{7,8} = [1; 3]$ $\quad R_{7,10} = [1; 3]$

$S_{8,2} = [0.5; 1]$ $\quad R_{8,9} = [0.5; 1]$

$R_{8,11} = [0; 2]$

$S_{9,1}= [0.25; 0.5]$ $\quad R_{9,8} = [0; 1.5]$ $\quad R_{9,10} = [0; 0.5]$

$S_{10,2} = [0.5; 1]$ $\quad R_{10,12} = [0.5; 1]$

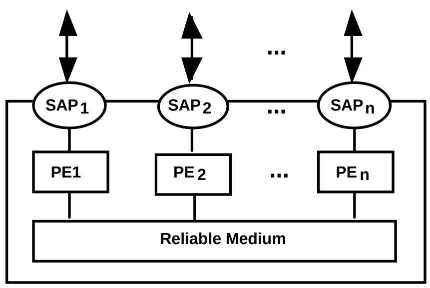
$S_{11,2}$= [0.5; 1] $R_{11,12}$ = [0.5; 1]

$S_{12,1}$ = [0.5; 1] $R_{12,1}$= [0.5; 1]

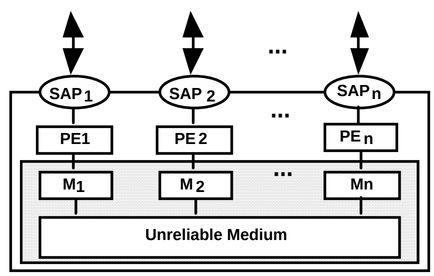
9. Deriving protocol with unreliable medium

9.1 Approach

When the medium is not reliable, two general approaches are thinkable. The first one consists of modifying the protocol entities PE_i obtained for reliable medium ([CL88]). The second, which is the one we have adopted, consists of inserting a new module M_i between each PE_i and the medium (fig.17.).



17.a. Realiable medium



17.b. Unreliable medium with modules

Figure 17. Addition of modules for an unreliable medium

The aim of each module M_i is to hide as much as possible the unreliability of the medium. The ideal would be that the unreliable medium combined with modules M_i is equivalent to a reliable medium. But in reality, it is not always possible

9.2. Classical examples

9.2.1. Transport Layer ([Ta90])

If the medium is made up of the three basic layers (physical, data linker and network), the added modules M_i can be the transport layer. If for instance the network is unreliable and generate N-Reset, then the transport protocol is of class 4.

9.2.2. "Alternating bit" protocol ([MB83])

If the medium can loose or garble messages, the modules M_i can for instance be the "alternating bit" protocol. On figure 18, there is an example of two communicating entities PE_1 and PE_2. Here, for simplicity, PE_1 is a sender and PE_2 is a receiver.

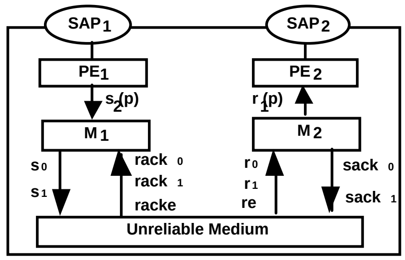


Figure 18. Alternating bit for an unreliable medium

s_i and r_i (i=0, 1) represent respectively the sending and receiving of an information frame which contains the last data block submitted by the user and the "alternating bit". Similarly, the operations $sack_i$ and $rack_i$ are the sending and receiving of an acknowledge frame which contains only a single bit. The operations re and and racke are a reception of a frame in error. Specifications of the medium, of the sender (M_1) and of the receiver (M_2) are given in [MB83], respectively on figures 10.a, 10.b and 10.f.

10. Conclusion

A method for deriving protocol for real-time applications is proposed in [KBD93]. In this paper, we improve and extend this method. We improve it by minimizing the number of exchanged messages. Consequences of this improvement are :

- conditions for existence of solutions are less strong. In some cases, approach in [KBD93] does not derive a protocol which respects a desired service, when the improved approach gives a solution.
- temporal requirements on derived protocols are less strong.

Extension of [KBD93] is done by considering an unreliable medium.

As in [KBD93], the time requirements can be calculated statically or dynamically. In the dynamic case, a method for exchanging complete temporal informations between entities is proposed. In this case, synchronization of local clocks is not necessary, so a global clock is not necessary. The dynamic case is interesting because the receiving protocol entities use more efficiently the time allocated to them to provide the service. In this paper, we give the same examples (sections 8) than those in [KBD93], but the derived protocols are not the same. Let's notice that the proposed algorithm can be useful in other areas than telecommunications (robotics ...) where several systems interact with each other to perform tasks in bounded delays. But there is a restriction : tasks are not concurrent.

At the present time, we are working for the two following improvements :

- considering concurrent tasks ,

- considering time requirements between events which are not consecutive.

References

[Al90] R. Alur, C. Courcoubetis and D. Hill, "Model checking for real-time systems." Proceedings of the 5th Symposium "Logic in computer Science", June 1990.

[BC79] W.A. Barrett and J.D. Couch, " Compiler Construction: Theory and Practice ", Publisher: Science Research Associates, Inc. 1979.

[BD91] B. Bertomieu and M.Diaz, "Modeling and verification of time dependant systems using Petri nets." IEEE Transactions of Software engineering, vol.17, No 3, March 1991.

[BG86] G.v. Bochmann and R. Gotzhein, "Deriving protocols specifications from service specifications." Proceedings du Symposium ACM SIGCOM '86, Vermont, USA, pp.148-156, 1986.

[CL88] P.Y.M. Chu and M.T.Liu, "Synthesizing Protocol specifications from service specifications in FSM model." Proceedings IEEE Computer Networking Symposium 1988.

[KHB92] C. Kant, T. Higashino and G.v. Bochmann, "Deriving protocol specifications from service specifications written in LOTOS." Rapport interne No 805, Département d'Informatique et de Recherche Opérationnelle. Faculté des arts et des sciences, Université de Montréal, January 1992.

[Ka91] M. Kapus Kolar, "Deriving protocol specifications from service specifications with heterogeneous timing requirements." Proceedings IEEE Int. Conf. on Software Engineering for real time systems, United-Kingdom, 1991.

[KBD93a] A. Khoumsi, G.v. Bochmann, and R. Dssouli, "Dérivation de spécifications de protocoles à partir de spécifications de services avec contraintes temporelles." Colloque Francophone pour l'ingénierie des protocoles (CFIP), Montréal, September 1993.

[KBD93] A. Khoumsi, G.v. Bochmann, and R. Dssouli, "Dérivation de spécifications de protocole à partir de spécifications de service avec des contraintes temps-réel." Soumis à la revue Réseaux et Informatique Répartie (RIR), Editions Hermès, Paris.

[KR91] M. Kapus Kolar and J. Rugelj, "Deriving protocol specifications from service specifications with simple relative timing requirements." Proceedings ISMM Int. Workshop on parallel computing, Italy, 1991.

[RDU85] C.V. Ramamoorthy, S.T. Dong and Y. Usuda, "An implementation of an automated protocol synthesizer (APS) and its application to the X21 protocol." IEEE Transactions on Sofware Engineering, Vol. SE-11, No 9, pp. 886-908, Sept. 1985.

[RBC92] N. Rico, G.v. Bochmann and O. Cherkaoui, "Model-Checking for real-time systems specified in LOTOS." CAV 1992.

[Si82] D. P. Sidhu, "Rules for synthesizing correct communication protocols." ACM SIGCOM comput. Commun., Rev. Vol. 12, No 1, pp.35-51, January 1982.

[Sid92] D. P. Sidhu, "Protocol design rules, Protocol specification, testing and verificaation." Ed. Sunshine C., North-Holland, pp.283-300, 1982.

[SP90] K.Saleh and R. Probert, "A service-based method for the synthesis of Communications protocols." International Journal of Mini and Microcomputers, Vol. 12, No 3, 1990.

[Ta90] A. Tanenbaum, "Réseaux : Architectures, protocoles, applications" InterÉditions, Paris 1990.

[ZWRCB80] Zafiropulo, C.H. West, H. Rudin, D.D. Cowan, and D. Brand, "Towards Analyzing and Synthesizing Protocols ", IEEE Transactions on Communications, Vol.28(4), April 1980, pp.651-661.