# Dérivation de spécifications de protocoles à partir de spécifications de services avec contraintes de temps

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

*Université de Montréal,*
*Faculté des arts et des sciences,*
*Département d'Informatique et*
*de Recherche Opérationnelle,*
*C.P. 6128, Succursale "A"*
*Montréal, (Quebec)*
*Canada H3C 3J7*

***Résumé.*** *Plusieurs méthodes de dérivation de spécifications de protocoles à partir des spécifications de services ont été développées dans [Boch86, Chu88, Kant92, Kap91, KapR91, Khen89, Rama85, Sid82, Sale90, Zafi80] . Les contraintes temporelles y sont abordées dans [Kap91, KapR91] mais avec des restrictions. Nous avons utilisé une approche analogue à celle présentée dans [Sale90], mais quelque peu modifiée, que nous avons étendue en y ajoutant des contraintes de temps d'une manière plus générale que dans [Kap91, KapR91]. Dans cet article nous proposons un algorithme de dérivation des contraintes de temps sur les entités de protocoles à partir des contraintes de temps sur le service. Nous traiterons les cas où les contraintes sur le protocole sont définies soit statiquement soit dynamiquement.*

*Mots-clés : spécifications de services et de protocole, dérivation de protocoles, contraintes temporelles statiques et dynamiques, contraintes temps-réel, synchronisation des horloges.*

***Abstract.*** *Several methods for deriving automatically protocol specifications from services specifications have been proposed in [Boch86, Chu88, Kant92, Kap91, KapR91, Khen89, Rama85, Sid82, Sale90, Zafi80]. Time requirements are considered in [Kap91, KapR91], but with restrictions . In this article, the approach of [Sale90] has been somewhat modified and extended by adding to it time requirements in a more general way than in [Kap91, KapR91]. We propose an algorithm for deriving time requirements on protocol entities from time requirements on the service. We consider the cases where the time requirements are static or dynamic.*

*Key-words : communication services, protocol specifications, deriving protocols, static and dynamic time requirements, real-time constraints, synchronizing clocks.*

## 1. Introduction

Dans [Sale90], à partir d'un automate à états finis spécifiant le service désiré, une méthode est proposée pour générer les automates spécifiant les protocoles au niveau

de chaque entité. L'ordonnancement des actions y est spécifié, mais les contraintes temporelles telles que les délais minimum et maximum entre deux actions successives ne sont pas considérées. Dans certaines applications, de telles spécifications ne sont pas suffisantes. Par exemple, elles ne permettent pas de spécifier que le temps séparant une demande de connexion et une confirmation (positive ou négative) soit inférieur à tmax. Plus généralement, il est très utile de pouvoir spécifier que le temps entre deux actions a1 et a2 consécutives soit dans un intervalle [tmin,tmax], où tmax peut être infini. Dans cet article, nous étendons la méthode présentée dans [Sale90]. A partir d'une spécification de service avec contraintes de temps, des spécifications des entités de protocoles avec contraintes de temps sont dérivées. Dans [Kap91, KapR91] le temps est aussi pris en compte pour la dérivation de protocoles spécifiés par un sous-ensemble de LOTOS. Mais [Kap91] suppose le temps de transit dans le medium négligeable, alors que [KapR91] l'estime par une valeur maximum. Dans les deux articles, les horloges locales des entités de protocoles sont supposées très précises et synchronisées, et ainsi équivalentes à une horloge globale à laquelle peuvent accéder toutes les entités.

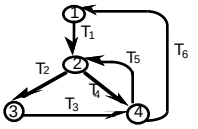
La suite de l'article est organisée comme suit. Dans la seconde section, nous présentons les règles de dérivation de protocoles sans contraintes de temps. Dans la section 3, le mode de spécification avec contraintes temporelles et la sémantique des intervalles de temps sont décrits. La section 4 définit les règles de calcul des contraintes temporelles pour les entités de protocoles à partir des contraintes temporelles sur le service. Nous traiterons les cas où celles-là sont calculées statiquement ou dynamiquement, sans faire les hypothèses de [Kap91,KapR91] présentées dans le paragraphe ci-dessus. Dans cette section une vérification est bien-sûr faite sur l'existence de solutions. Dans la section 5, nous présentons les différentes étapes de la procédure de dérivation des spécifications de protocoles avec contraintes temporelles. La section 6 est consacrée à des exemples. Et enfin nous concluons.

## 2. Règles de dérivation des entités de protocoles sans contraintes de temps

### *2.1. Spécification de service*

Le service désiré est décrit par une FSM, que nous notons SP_S, qui spécifie les séquences de primitives de service (PS) que l'on voudrait observer au niveau des différents points d'accès de service (PAS). A chaque PAS correspond une entité de protocole (EP), nous ne distinguerons pas par la suite une EP et le PAS correspondant. Les transitions de SP_S sont définies par trois paramètres (fig.1) qui sont: - la primitive de service E ayant entraîné le changement d'état,

- un numéro a identifiant l'entité ou le PAS où a lieu cette primitive de service E, une telle entité est notée $EP_a$ ,
- un numéro p identifiant la transition, une telle transition est notée $T_p$=(E,a,p).




$T_1$ = (A, 1, 1) $T_2$ = (B, 2, 2) $T_3$ = (C, 3, 3)
$T_4$ = (E, 4, 4) $T_5$ = (F, 1, 5) $T_6$ = (D, 2, 6)

**Figure 1**. *Spécification de service*

Deux transitions différentes de SP_S auront leurs troisièmes paramètres p différents. Ceux-ci sont utilisés dans les messages qui transiteront entre les différentes EP. Une transition sera donc désignée par (E,a,p) et signifiera que la primitive E a lieu sur $EP_a$. Comme dans [Sale90], pour un état e de SP_S, out(e) et in(e) désignent respectivement les ensembles des PAS correspondant aux transitions sortantes et entrantes. Exemple : sur la figure 1, in(2) = $\{PAS_1\}$, et out(2)=$\{PAS_2,PAS_4\}$.

### *2.2. Procédure de dérivation des spécifications de protocoles*

La dérivation consiste à générer autant de FSM que de EP. Chacune des FSM que nous notons $SP\_P_i$, spécifie les séquences d'actions qui ont lieu sur l'entité $EP_i$. Pour la réalisation du service désiré, les différentes entités $EP_i$ devront communiquer entre elles à travers un medium que nous supposerons parfait. Le principe de dérivation est plutôt simple: lorsque dans le service deux primitives de services A et B consécutives sont exécutées par des EP différentes, soit $EP_i$ et $EP_j$, alors :
- après exécution de A par $EP_i$, celle-ci envoie un message m à l'entité $EP_j$
- après réception du message m par $EP_j$, celle-ci exécute B

Les règles de dérivation que nous avons adoptées sont analogues à celles de [Sale90] mais quelque peu modifiées. En effet, nous ne distingons ni le sens des primitives de service entre le fournisseur et l'utilisateur du service, ni l'état initial des autres états. La syntaxe d'échange des messages a également été modifiée. La procédure de dérivation peut être décomposée en trois étapes.

***Etape 1*** **:** A partir de SP_S, nous générons pour chaque $EP_i$ l'automate $SP\_S_i$ obtenu en remplaçant toute transition (E,a,p) de SP_S par :
- (#,a,p) si a≠i (c'est-à-dire si l'action E a lieu sur une EP ≠ $EP_i$) ,
- (E,p) si a=i .

Si nous prenons l'exemple de la figure 1, nous obtenons les $SP\_S_i$ de la figure 2.

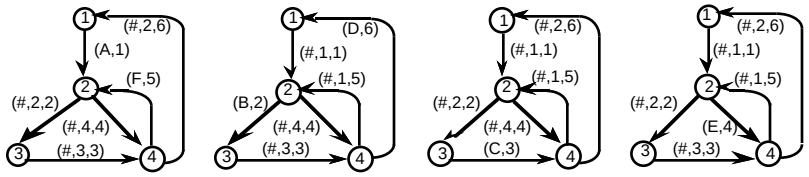
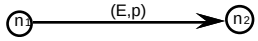
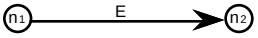
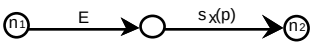



**Figure 2**. *Les $SP\_S_i$ obtenus à l'étape 1*

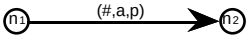
***Etape 2*** **:** A partir de chaque $SP\_S_i$, nous générons $SP\_SP_i$ selon les règles de dérivation suivantes :

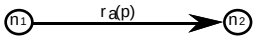
- *Pour une transition (E,p)*: 

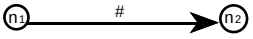

*Cas a*: si out(n2)=$\{PAS_i\}$ la transition devient : 


*Cas b*: si out(n2)≠$\{PAS_i\}$ la transition devient : 


Avec x= out(n2) - $\{PAS_i\}$, et $s_x(p)$ signifie envoi de message paramétré par p aux PAS appartenant à x

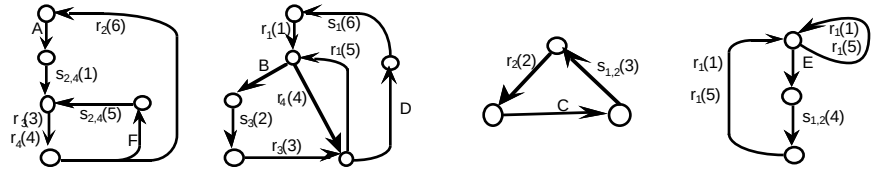
*-Pour une transition (#,a,p):* 


*Cas c* : si out(n2) contient $PAS_i$, la transition devient : 


*Cas d :*si out(n2) ne contient pas $PAS_i$, la transition devient: 


Avec $r_a(p)$ signifiant réception de message paramétré par p en provenance de $EP_a$.

***Etape 3*** **:** Les transitions # des $SP\_SP_i$ sont considérées spontanées et sont supprimées par projection pour obtenir les spécifications de protocoles $SP\_P_i$ . Un algorithme de suppression des # est donné dans [Barr79]. Intuitivement, soit $A_\#$ un automate contenant des transitions # et spécifiant un système $\mathcal{A}$, et soit A l'automate obtenu par suppression des #. Si un observateur externe peut détecter toutes les transitions sauf #, alors A est la spécification de $\mathcal{A}$ tel qu'il est perçu par l'observateur. Pour notre exemple, nous obtenons:




**3.a.** *SP_P1* **3.b.** *SP_P2* **3.c.** *SP_P3* **3.d.** *SP_P4*

**Figure 3**. *Spécifications de protocoles obtenues.*

### *2.3. Correction et unicité de la solution*

Le terme "correction" signifie ici le fait d'être correct. On démontre dans [Sale90] que les $SP\_P_i$ obtenus sont *sémantiquement et syntaxiquement correctes*. Leur sémantique est correcte car les entités dérivées fournissent le service désiré. Leur syntaxe est correcte car ces entités ne contiennent ni blocage(deadlock), ni boucle infinie (livelock), et aucune erreur de réception non spécifiée n'est possible. Précisons aussi que la solution est *unique* et ne dépend pas de l'ordre dans lequel les transitions sont traitées.

### *2.4. Sémantique du choix entre plusieurs actions*

Lorsqu'un état n est atteint par une transition $T_p$=(E,a,p) et que plusieurs transitions $T_{pi}$=(Ei,ai,pi) sont ensuite possibles (fig. 4), un choix doit bien-sûr être fait. Il faut alors considérer les trois cas ci-devant.

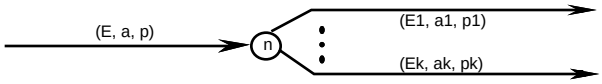



**Figure 4**. *Choix entre plusieurs actions.*

*Cas 1*: Au moins deux actions Ei sont sur des entités $EP_{ai}$ différentes (il existe ai et aj tels que ai≠aj). Le choix est alors fait par l'entité $EP_a$ qui exécute l'action E, et indiqué dans le message qu'elle envoie aux différentes entités $EP_{ai} \neq EP_a$. Ce cas est important puisque les règles de dérivation l'ont implicitement considéré. En effet, les différentes $EP_{ai}$ ne communiquant pas entre elles, le

choix ne peut être fait que par l'entité $EP_a$ qui leur transmet sa "décision".

*Cas 2*: Toutes les actions Ei sont exécutées sur la même $EP_b \neq EP_a$, le choix peut alors être fait par $EP_a$ ou par $EP_b$.

*Cas 3*: Toutes les actions Ei sont exécutées sur $EP_a$, le choix est fait par $EP_a$.

## 3. Spécifications de service avec contraintes de temps

Sur une spécification de service avec contraintes de temps (SP_ST), chaque transition est définie par :

- les trois paramètres que nous avons présentés dans la section 2.1 ,
- un ensemble $C_p$ d'intervalles de temps, où p est le numéro identifiant la transition .

Une transition est donc définie par $T_p=(E,a,p,C_p)$. L'exécution d'une transition $T_p$ signifiera par la suite l'exécution de l'action E de la transition $T_p$ par l'entité $EP_a$. Considérons pour un état n de SP_ST, ses k transitions entrantes $T_{pi}$, et une de ses transitions sortantes $T_p$ (fig.5). La représentation de la figure 5 nous suffit pour définir la sémantique de l'ensemble $C_p$ de la transition sortante. Celui-ci contient autant d'intervalles de temps qu'il y a de transitions entrantes sur le noeud n. Chaque intervalle, que nous désignons par $T_{pi,p}=[T^{mi}_{pi,p}; T^{ma}_{pi,p}]$ pour i allant de 1 à k,est donc associé à une des transitions entrantes $T_{pi} = (Ei,ai,pi,C_{pi})$. La sémantique est ([Alur90,Bert91,Rico92]): Pour chaque transition entrante $T_{pi}$, la transition $T_p$ doit être exécutée dans l'intervalle $T_{pi,p}$ après que l'état n ait été atteint par la transition $T_{pi}$, à moins qu'une autre transition ne soit tirée entre temps. Autrement dit, la transition $T_p$ sera tirée dans l'intervalle $T_{pi,p}$ après l'exécution de la transition $T_{pi}$, sauf si une autre transition sortante du noeud n est tirée avant $T^{ma}_{pi,p}$. Nous déduisons de tout ceci, que si l'état n est l'état initial, alors les différents intervalles de $C_p$ doivent être égaux. Remarquons que $T_p$ est une transition identifiée par p, alors que $T_{p,q}$ est l'intervalle contenant le temps entre les transitions $T_p$ et $T_q$.

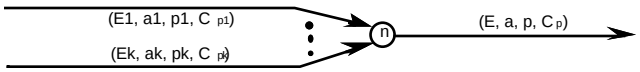



**Figure 5.** *Transitions entrantes sur un noeud de SP_ST*

## 4. Calcul des contraintes pour les entités de protocoles

Lorsqu'une action E d'une transition $T_p = (E,a,p,C_p)$ est exécutée sur une entité $EP_a$, un message est alors envoyé par $EP_a$ à toutes les autres entités devant exécuter des actions consécutives à E.

### *4.1. Notations et approche du problème de calcul de contraintes temporelles (PCCT)*

Pour la dérivation des contraintes de temps sur les entités de protocoles, il faut considérer, à la fois sur un noeud donné, une de ses transitions entrantes et toutes ses transitions sortantes telles qu'au moins une de ces dernières est exécutée sur une entité de protocole différente de celle de la transition entrante. Autrement dit, sur la figure 6, pour les aj ( j=1 à k), il existe au moins un aj tel que aj≠a.

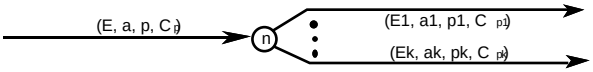



**Figure 6**. *Transitions sortantes sur un noeud de SP_ST*

Soit alors une transition $T_p$ sur une entité $EP_a$ suivie par plusieurs transitions $T_{pj}$, j=1 à k, sur $EP_{aj}$. Après $T_p$, l'entité $EP_a$ doit envoyer un message aux $EP_{aj}$ ($aj \neq a$). Et après réception du message par une entité $EP_{aj}$ ($aj \neq a$), celle-ci peut exécuter $T_{pj}$ si cette dernière a été choisie parmi les différentes transitions sortantes. Nous pouvons représenter ces actions pour chaque $T_{pj}$ (exécuté sur $aj \neq a$) en fonction du temps par la figure 7. Sur cette figure, x est l'ensemble des entités $EP_{aj}$ ( j=1 à k et $aj \neq a$). La contrainte temporelle de service impose à $t^j$ d'appartenir à $T_{p,pj}=[T^{mi}_{p,pj}; T^{ma}_{p,pj}]$. $t_1$ est le temps séparant l'exécution de $T_p$ par $EP_a$ et l'émission d'un message aux autres $EP_{aj}$ (j=1 à k et $aj \neq a$). $t^j_2$ est le temps de transit du message dans le medium pour aller de $EP_a$ à $EP_{aj}$ ($aj \neq a$). Nous définissons pour chaque $EP_{aj}$, le plus petit intervalle de temps que nous connaissons contenant toujours $t^j_2$. Cet intervalle qui dépend de $EP_a$ et $EP_{aj}$ est noté $M_{a,aj}=[M^{mi}_{a,aj},M^{ma}_{a,aj}]$. $t^j_3$ est le temps entre l'arrivée à $EP_{aj}$ du message envoyé par $EP_a$ et l'exécution de $T_{pj}$.

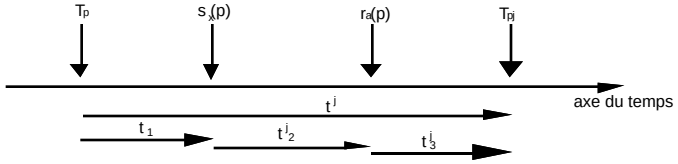



**Figure 7**. *Représentation dans le temps de deux actions $T_p$ et $T_{pj}$ consécutives exécutées sur des entités différentes*

Si la transition sortante $T_{pj}$ choisie est sur $EP_a$ (càd aj=a), alors $EP_a$ envoie un message aux autres entités et exécute ensuite $T_{pj}$. Autrement dit, *si aj=a, alors $t^j_2=0$ et il n'y a pas d'action $r_a(p)$*. Dans ce cas, nous prenons $M_{a,aj} = [0; 0]$, et $t^j_3$ est le temps entre l'envoi du message aux autres entités et l'exécution de $T_{pj}$.

Le but de la dérivation des contraintes temporelles sur les entités de protocoles est le suivant. Connaissant les contraintes $t^j \in T_{p,pj}$ et $t^j_2 \in M_{a,aj}$, pour j= 1 à k, nous devons calculer la contrainte sur $t_1$ et celles sur $t^j_3$ qui assurent les vérifications des contraintes de service $t^j \in T_{p,pj}$. Ces contraintes seront sous la forme $t_1 \in S_p=[S^{mi}_p,S^{ma}_p]$, et $t^j_3 \in R_{p,pj}=[R^{mi}_{p,pj}, R^{ma}_{p,pj}]$ pour j =1 à k. Le but est donc de calculer les $S_p$ et $R_{p,pj}$ connaissant les $T_{p,pj}$ et $M_{a,aj}$.

Nous utiliserons également les notations suivantes :

- s(p)=a si $T_p$ = (E,a,p), s est donc une application qui associe à l'identificateur de la transition $T_p$, l'identificateur de l'entité de protocole exécutant $T_p$. s(p) sera noté sp par la suite, et nous avons donc $M_{a,aj} = M_{sp,spj}=[M^{mi}_{sp,spj}, M^{ma}_{sp,spj}]$.
- $V^{mi}_p$ et $V^{ma}_p$ sont des paramètres appartenant à [0,1] définis pour toute transition $T_p$ suivie par au moins une transition sur une autre entité. Ils sont utilisés pour le choix d'une solution parmi une infinité. Si nous obtenons, comme nous le verrons par la suite, pour $S_p=[S^{mi}_p,S^{ma}_p]$ la contrainte $S^{ma}_p \in [a,b]$, nous choisirons $S^{ma}_p=a+V^{ma}_p*(b-a)$. De même, si nous obtenons $S^{mi}_p \in [c,d]$, nous

choisirons $S^{mi}_p = c + V^{mi}_p * (d-c)$.

- Nous définissons l'addition et la soustraction de deux intervalles [a, b] et [c, d] par [a, b] + [c, d] = [a + c, b + d] , et [a, b] - [c, d] = [a - c, b - d].

Pour résumer, les entrées du PCCT pour les entités de protocoles sont :

- $T_{p,q}$ et $M_{sp,sq}$ pour toute paire de transitions $T_p$ et $T_q$ consécutives.
- $V^{mi}_p$ et $V^{ma}_p$ qui sont des paramètres appartenant à [0,1] définis pour toute transition $T_p$ suivie par au moins une transition sur une autre entité. Ils sont utilisés pour le choix d'une solution particulière parmi une infinité.

Les solutions du PCCT sont :

- $S_p$ pour toute transition $T_p$ suivie par au moins une transition sur une autre EP.
- $R_{p,q}$ pour toute paire de transitions $T_p$ et $T_q$ consécutives .

Nous verrons qu'il existe des conditions sur les premières données du PCCT ($T_{p,q}$ et $M_{p,q}$) pour que les solutions existent. S*i toutes les transitions $T_{pj}$ qui suivent une transition $T_p$ sont sur la même entité que celle-ci, alors $R_{p,pj} = T_{p,pj}$.*

### *4.2. Condition d'existence des solutions*

Pour les transitions représentées sur les figures 6 et 7, pour que le service obtenu soit inclus dans le service désiré, nous déduisons de la figure 7 :

$$\text{pour } j=1 \text{ à } k : t^j \in T_{p,pj} \text{ implique } t_1 + t^j_2 + t^j_3 \in T_{p,pj} \qquad [1]$$

Si $S_p$ et $R_{p,pj}$ sont tels que, pour tout $t_1 \in S_p$ et $t^j_3 \in R_{p,pj}$ et $t^j_2 \in M_{sp,spj}$ , la condition [1] est vérifiée, alors il faut et il suffit que :

$$\text{pour } j=1 \text{ à } k : \quad S_p + M_{sp,spj} + R_{p,pj} \subseteq T_{p,pj} \qquad [2]$$

C'est-à-dire :

$$\text{pour } j=1 \text{ à } k : \quad S^{mi}_p + M^{mi}_{sp,spj} + R^{mi}_{p,pj} \geq T^{mi}_{p,pj} \qquad [3]$$

$$S^{ma}_p + M^{ma}_{sp,spj} + R^{ma}_{p,pj} \leq T^{ma}_{p,pj} \qquad [4]$$

Une condition nécessaire et suffisante pour que le service fourni par le protocole puisse satisfaire le service désiré est donc :

$$\text{pour } j=1 \text{ à } k : \quad T^{ma}_{p,pj} - M^{ma}_{sp,spj} \geq \sup(T^{mi}_{p,pj} - M^{mi}_{sp,spj}; 0) \qquad [5]$$

La condition [5] doit bien-sûr être vérifiée pour tous les noeuds de SP_ST. Et pour chaque noeud, elle doit être vérifiée pour toutes les transitions entrantes. Pour chaque noeud et une de ses transitions entrantes $T_p$, la résolution du PCCT consiste ainsi à: - vérifier la condition [5],

- si la vérification est positive alors :
  - * déterminer $S_p$
  - * déterminer les $R_{p,pj}$ pour toutes les transitions sortantes $T_{pj}$

Signalons que si [5] est vérifiée, le nombre de solutions est infini. Nous montrerons par la suite comment une solution particulière est choisie à l'aide des paramètres $V^{ma}_p$ et $V^{mi}_p$. Comme les équations et inéquations seront toujours

données pour un noeud donné et une de ses transitions entrantes, notre étude se généralise implicitement à tous les noeuds et à toutes leur transitions entrantes. Pour résoudre le PCCT, nous considérons les trois cas suivants :

*Cas 1* les messages transmis par les EP ne contiennent pas d'information temporelle,
*Cas 2* : les EP émettent leurs messages en y ajoutant une information temporelle,
*Cas 3* : En plus de l'information de temps de l'EP, le medium ajoute dans le message une seconde information temporelle qui est une estimation de la durée de transit du message dans le medium. C'est ce cas, traité en 4.3.3, qui permet à une entité réceptrice d'avoir une information temporelle complète *sans qu'une horloge globale soit nécessaire*.

### *4.3. Résolution du PCCT*

*4.3.1. Cas 1: Messages sans informations de temps (cas statique)*

Une entité $EP_{aj}$ (aj ≠ a) n'a aucune information de temps dans le message qu'elle reçoit de $EP_a$ (fig. 6). La seule connaissance qu'elle a est que l'action $T_p$ a eu lieu dans un temps appartenant à la somme d'intervalles $S_p + M_{sp,spj}$ avant réception du message. Rappelons que si aj=a il suffit de prendre $M_{sp,spj}=[0; 0]$.

Résolution : La condition [4] entraîne :

$$S^{ma}{}_p \in [0; \min_{j=1 \text{ à } k} (T^{ma}{}_{p,pj} - M^{ma}{}_{sp,spj})] \quad [6]$$

Et [3] et [4] impliquent alors : $$S^{mi}{}_p \in [\sup(U, 0); S^{ma}{}_p] \quad [7]$$

avec $$U = \max_{j=1 \text{ à } k} (S^{ma}{}_p + (M^{ma}{}_{sp,spj} - M^{mi}{}_{sp,spj}) - (T^{ma}{}_{p,pj} - T^{mi}{}_{p,pj})) \quad [8]$$

Nous avons ensuite les solutions sur $R_{p,pj} = [R^{mi}{}_{p,pj} ; R^{ma}{}_{p,pj}]$ :

Soit donc pour j=1 à k : $$R^{ma}{}_{p,pj} = T^{ma}{}_{p,pj} - M^{ma}{}_{sp,spj} - S^{ma}{}_p \quad [9]$$

$$R^{mi}{}_{p,pj} = \sup(T^{mi}{}_{p,pj} - M^{mi}{}_{sp,spj} - S^{mi}{}_p ; 0) \quad [10]$$

Le choix d'une solution particulière pour $S^{ma}{}_p$ et $S^{mi}{}_p$ vérifiant [6] et [7], et donc pour $R^{ma}{}_{p,pj}$ et $R^{mi}{}_{p,pj}$ vérifiant [9] et [10], peut être fait à l'aide des paramètres $V^{ma}{}_p$ et $V^{mi}{}_p$ appartenant à [0,1]. Ces derniers étant choisis, nous prenons :

$$S^{ma}{}_p = V^{ma}{}_p * \min_{j=1 \text{ à } k} (T^{ma}{}_{p,pj} - M^{ma}{}_{sp,spj}) \quad [11]$$

$$S^{mi}{}_p = \sup(U, 0) + (S^{ma}{}_p - \sup(U, 0)) * V^{mi}{}_p) \quad [12]$$

où U est défini dans [8]. On peut aisément vérifier que [11] et [12] respectent [6] et [7], et que le service obtenu satisfait le service désiré. Il est préférable de choisir $V^{ma}{}_p$ le plus petit possible et $V^{mi}{}_p$ le plus grand possible. Ceci implique d'avoir $S^{ma}{}_p$ et $S^{mi}{}_p$ les plus petits et proches possibles. $R^{ma}{}_{p,pj}$ et $R^{mi}{}_{p,pj}$ seront ainsi les moins contraints possible, et les entités réceptrices disposeront du maximum de temps possible pour satisfaire le service.

*4.3.2. Cas 2: Messages reçus avec une information temporelle (cas dynamique)*

Une fois la transition $T_p$ exécutée par $EP_a$ (fig. 6), celle-ci envoie un message au

bout d'un temps $S^m$ aux autres entités $EP_{aj}$ (j = 1 à k et aj ≠ a). La valeur $S^m$ est contenue dans le message. L'information que possède $EP_{aj}$ se précise par rapport au cas précédent (4.3.1), elle sait que l'action $T_p$ a eu lieu dans un temps appartenant à la somme d'intervalles $[S^m, S^m] + M_{sp,spj}$ avant l'arrivée du message. Elle peut alors choisir $R_{p,pj}$ dynamiquement en fonction de $S^m$, et mieux exploiter le temps qui lui reste avant l'exécution de $T_{pj}$.

Résolution: la condition [4] implique toujours que $S^{ma}{}_p$ vérifie la condition [6], et $S^{mi}{}_p$, moins contraint que dans le cas précédent, peut alors être pris tel que :

$$S^{mi}{}_p \in [0 ; S^{ma}{}_p] \qquad [13]$$

Si $S^m$, appartenant à $[S^{mi}{}_p; S^{ma}{}_p]$, est le temps au bout duquel le message est envoyé, l'entité réceptrice le connaîtra et pourra choisir :

$$\text{pour j=1 à k :} \quad R^{ma}{}_{p,pj}(S^m) = T^{ma}{}_{p,pj} - M^{ma}{}_{sp,spj} - S^m \qquad [14]$$

$$R^{mi}{}_{p,pj}(S^m) = \sup(T^{mi}{}_{p,pj} - M^{mi}{}_{sp,spj} - S^m;\ 0) \qquad [15]$$

Comme dans 4.3.1 le choix d'une solution particulière est fait à l'aide des paramètres $V^{ma}{}_p$ et $V^{mi}{}_p$. $S^{ma}{}_p$ est donc résolu toujours à l'aide de l'équation [11] et

$$S^{mi}{}_p = S^{ma}{}_p * V^{mi}{}_p \qquad [16]$$

On peut vérifier que le service obtenu satisfait le service désiré. L'information $S^m$ permet à une entité réceptrice $EP_{aj}$ de mieux exploiter le temps qui lui est réservé pour satisfaire le service que dans le cas statique. En effet, l'intervalle de temps $R_{p,pj}(S^m)$ ([14] et [15]) est moins contraignant que l'intervalle $R_{p,pj}$ ([9] et [10]), puisque $R_{p,pj}$ est strictement inclus dans $R_{p,pj}(S^m)$.

*4.3.3. Messages reçus avec deux informations temporelles (second cas dynamique)*

En plus de l'information $S^m$, le medium ajoute dans le message envoyé l'information $M^m$, qui représente le temps passé par le message dans le medium. Une entité réceptrice $EP_{aj}$ connait donc le temps écoulé entre l'action $T_p$ par $EP_a$ et l'arrivée du message. Ce temps est $S^m+M^m$. Cette fois-ci encore $R_{p,pj}$ est choisi dynamiquement, en fonction de $S^m$ et $M^m$, par l'entité $EP_{aj}$. La résolution de $S_p=[S^{mi}{}_p; S^{ma}{}_p]$ est identique à celle de 4.3.2. $R^{ma}{}_{p,pj}$ et $R^{mi}{}_{p,pj}$ sont choisis tels que:

$$\text{pour j=1 à k :} \quad R^{ma}{}_{p,pj}(S^m, M^m) = T^{ma}{}_{p,pj} - M^m - S^m \qquad [17]$$

$$R^{mi}{}_{p,pj}(S^m, M^m) = \sup(T^{mi}{}_{p,pj} - M^m - S^m;\ 0) \qquad [18]$$

Précisons que si aj=a alors $M^m$ doit être pris égal à 0. On peut toujours vérifier que dans ce second cas dynamique, le service désiré est toujours satisfait par le protocole. L'information $M^m$ supplémentaire permet à l'entité réceptrice d'encore mieux exploiter le temps qui lui est réservé pour satisfaire le service que dans le premier cas dynamique. En effet, $R_{p,pj}(S^m)$ ([14] et [15]) est strictement inclus dans $R_{p,pj}(S^m, M^m)$ ([17] et [18]). Dans les deux cas dynamiques, une synchronisation entre les horloges des différentes entités de protocoles n'a pas été nécessaire pour l'échange des informations temporelles entre ces entités. *Le second cas dynamique*

*est intéressant, car il permet l'échange d'informations temporelles complètes qui ne nécessitent pas l'utilisation d'une horloge globale .*

*4.3.4. Temps de transit dans le medium*

Précisons aussi , pour le second cas dynamique, que le temps $M^m$ de transit dans le medium est une estimation et non pas une valeur très précise. En effet, si le message parcourt plusieurs noeuds avant d'arriver à destination, $M^m$ comprend les estimations des temps :

* de transmission et de propagation entre les différents noeuds adjacents,
* de séjour dans les noeuds (traitements et surtout attentes dans les files).

Pour cette raison, des paramètres $\alpha$ et $\beta$ de sécurité peuvent être utilisés dans les équations [17] et [18] qui deviennent:

pour j=1 à k et aj ≠ a: $R^{ma}_{p,pj}(S^m, M^m) = T^{ma}_{p,pj} - M^m - S^m - \beta$ [19]

$R^{mi}_{p,pj}(S^m, M^m) = \sup(T^{mi}_{p,pj} - M^m - S^m + \alpha ; 0)$ [20]

Ceci revient à estimer la durée de transit dans l'intervalle $[ M^m - \alpha ; M^m + \beta ]$. Dans la section suivante, nous présentons les quatre étapes qui composent la procédure de dérivation des spécifications de protocoles.

## 5. Dérivation de spécifications de protocoles avec contraintes de temps

***Etape 0*** : A partir de la spécification SP_ST, nous générons SP_STT défini ci-devant. En nous basant toujours sur les figures 6 et 7 :

- Dans *le cas statique*, une entité réceptrice $EP_{aj}$ doit connaître l'intervalle constant $R_{p,pj}$ (calculé par [9] et [10]), qui représente l'intervalle de temps qui lui est réservé depuis réception de message pour l'exécution de $T_{pj}$.

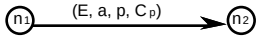
- Dans *le premier cas dynamique*, une entité réceptrice $EP_{aj}$ doit connaître l'intervalle constant $X_{p,pj} = T_{p,pj} - M_{sp,spj}$ et le paramètre $S^m$ reçu dans le message pour calculer dynamiquement $R_{p,pj}$ (par les équations [14] et [15]).

- Dans *le second cas dynamique*, $EP_{aj}$ doit connaître l'intervalle constant $T_{p,pj}$ et les paramètres $S^m$ et $M^m$ reçus dans le message pour calculer dynamiquement $R_{p,pj}$ (par les équations [17] et [18]).

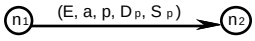
Nous déduisons de ceci que SP_STT est obtenu à partir de SP_ST par :

* association des intervalles de temps $S_p$ aux transitions $T_p$ qui sont suivies par au moins une transition sur une autre entité (fig. 6).
* substitution des intervalles de temps $T_{p,pj}$ par les intervalles $R_{p,pj}$ dans le cas statique, ou par les intervalles $X_{p,pj}$ dans le premier cas dynamique. La substitution n'est pas faite dans le second cas dynamique.

Pour obtenir SP_STT toute transition $T_p$ de SP_ST :




est donc remplacée par la transition :



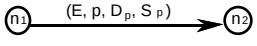

$D_p$ étant l'ensemble des intervalles $R_{pi,p}$ dans le cas statique, $X_{pi,p}$ dans le premier cas dynamique, et $T_{pi,p}$ dans le second cas dynamique. Où les pi identifient

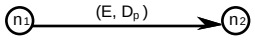
les transitions entrantes de n1. Précisons que les intervalles appartenant à $D_p$ n'ont pas la même sémantique dans les trois cas. Dans le cas statique, $R_{pi,p}$ est une contrainte temporelle constante, alors que dans les deux autres cas, $X_{pi,p}$ et $T_{pi,p}$ sont des intervalles qui sont utilisés pour le calcul dynamique de la contrainte temporelle sur $T_p$ lorsque celle-ci succède à la transition $T_{pi}$.

La complexité de l'algorithme de génération de SP_STT est en n*e*s avec :
- n : nombre d'états de SP_ST
- e : nombre maximum de transitions entrantes par état (ou noeud)
- s : nombre maximum de transitions sortantes par état

***Etape 1*** **:** A partir de SP_STT, nous générons pour chaque $EP_i$ l'automate $SP\_ST_i$ obtenu en remplaçant toute transition $(E,a,p,D_p,S_p)$ par : $(\#,a,p)$ si $a \neq i$, et par $(E,p,D_p,S_p)$ si $a=i$.

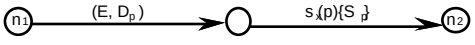
***Etape 2*** : A partir de chaque $SP\_ST_i$, nous générons les $SP\_SPT_i$ selon les règles de dérivation suivantes :

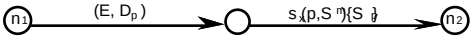
- *Pour les transitions $(E,p,D_p,S_p)$* de $SP\_ST_i$ soit : 

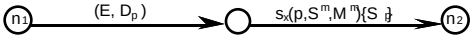

*Cas a* : si $out(n2) = \{PAS_i\}$ la transition devient : 


Dans ce cas a, l'intervalle $S_p$ n'est en fait pas défini.

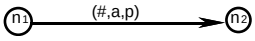
*Cas b* : si $out(n2) \neq \{PAS_i\}$ nous obtenons :

* cas statique : 

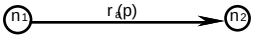

* premier cas dynamique : 

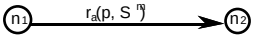

* second cas dynamique : 

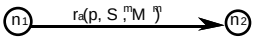

Avec x = out(n2) - $\{PAS_i\}$, $s_x(p)$ signifiant l'émission du message paramétré par p vers les entités de x (cas statique). Dans $s_x(p, S^m)$, le message émis est paramétré par p et $S^m$ (premier cas dynamique), et dans $s_x(p,S^m,M^m)$ le message émis contient les trois paramètres p, $S^m$ et $M^m$. $\{S_p\}$ spécifie que $s_x(p)$ doit se faire au bout d'un temps appartenant à $S_p$ après l'action qui la précède.

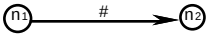
*-Pour les transitions (#,a,p)* : 


*Cas c* : si out(n2) contient $PAS_i$, la transition devient :

* cas statique : 


* premier cas dynamique : 


* second cas dynamique : 


*Cas d* : si $PAS_i$, n'est pas dans out(n2), la transition devient : 


Avec $r_a(p)$ signifiant réception de message paramétré par p en provenance de $EP_a$. Dans $r_a(p,S^m)$ le message reçu est paramétré par p et $S^m$ (premier cas dynamique), alors que dans $r_a(p, S^m, M^m)$ le message reçu contient les paramètres p, $S^m$ et $M^m$.

***Etape 3*** **:** Les transitions #, qui sont considérées spontanées, sont supprimées par projection et nous obtenons les spécifications de protocoles avec contraintes de temps désignées par $SP\_PT_i$.

## 6. Exemples

### *6.1. Cas statique*

Nous considérons l'exemple 1 (fig. 1) auquel nous avons ajouté les contraintes temporelles (fig.8). Nous traitons ici le cas 1 où les contraintes sont calculées statiquement. Mais il n'y a aucun problème pour l'étendre au cas dynamique. Sur la figure 8, $C_1=\{T_{6,1}\}$, $C_2=\{T_{1,2},T_{5,2}\}$, $C_3=\{T_{2,3}\}$, $C_4=\{T_{1,4},T_{5,4}\}$, $C_5=\{T_{3,5},T_{4,5}\}$, et $C_6=\{T_{3,6},T_{4,6}\}$. Avec $T_{6,1}=[3,6]$, $T_{1,2}=[5,10]$, $T_{5,2}=[5,10]$, $T_{2,3}=[3,7]$, $T_{1,4}=[4,9]$, $T_{5,4}=[4,9]$, $T_{3,5}=[3,7]$, $T_{4,5}=[3,7]$, $T_{3,6}=[2,6]$, $T_{4,,6}=[2,6]$.

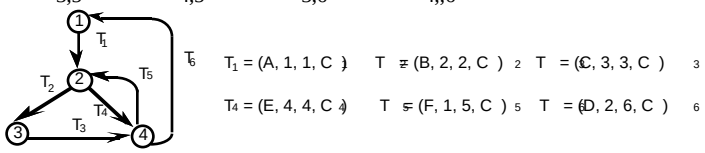
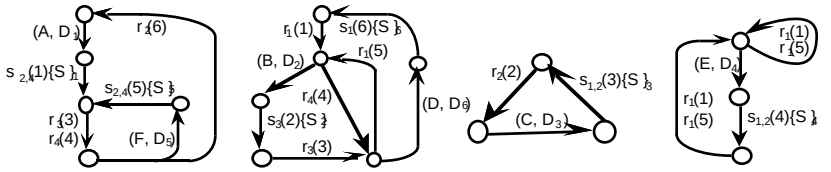



**Figure 8**. *Spécification de service avec contraintes de temps*

Nous avons pris pour le medium $M_{u,v}$ = [2,4] pour tout (u,v), et enfin les paramètres $V^{ma}{}_p$ et $V^{mi}{}_p$ suivants : $V^{ma}{}_1$= 0.6 , $V^{ma}{}_2$= 0.5 , $V^{ma}{}_3$= 0.65 , $V^{ma}{}_4$= 0.5 , $V^{ma}{}_5$= 0.7 , $V^{ma}{}_6$= 0.5, $V^{mi}{}_1$= 0.466, $V^{mi}{}_2$= 0.33, $V^{mi}{}_3$= 0.53 , $V^{mi}{}_4$= 0 , $V^{mi}{}_5$=0.5, $V^{mi}{}_6$= 0.5. Les spécifications de protocoles avec contraintes de temps dérivées sont représentées sur la figure 9, avec $D_1=\{R_{6,1}\}$, $D_2=\{R_{1,2},R_{5,2}\}$, $D_3=\{R_{2,3}\}$, $D_4=\{R_{1,4}, R_{5,4}\}$, $D_5=\{R_{3,5}, R_{4,5}\}$, $D_6=\{R_{3,6}, R_{4,6}\}$.

A partir des équations [11], [12], [9] et [10], nous calculons :
$S_1$ = [1, 3], $S_2$ = [0.5, 1.5], $S_3$ = [0.7, 1.3], $S_4$ = [0, 1] , $S_5$ = [2, 3.5], $S_6$ = [0.5, 1], $R_{6,1}$ = [0.5, 1], $R_{1,2}$ = [1.6, 3], $R_{5,2}$ = [1, 2.5], $R_{2,3}$ = [0.5, 1.5], $R_{1,4}$ = [0.6, 2], $R_{5,4}$ = [0, 1.5], $R_{3,5}$ = [0.3, 1.7], $R_{4,5}$ = [1, 2], $R_{3,6}$ = [0, 0.7], $R_{4,6}$ = [0, 1] .




**9.a.** *SP_PT1* **9.b.** *SP_PT2* **9.c.** *SP_PT3* **9.d.** *SP_PT4*

**Figure 9**. *Spécifications de protocoles avec contraintes de temps*

### *6.2. Premier cas dynamique*

Nous avons choisi un exemple très simple, où deux entités de protocole $EP_1$ et $EP_3$ assurent le service spécifié par l'automate de la figure 10, avec $C_1$ = $\{T_{2,1}\}$ et $C_2$ = $\{T_{1,2}\}$. Nous prenons $T_{2,1}$=[5,10] et $T_{1,2}$=[4,8], $M_{1,3}$=$M_{3,1}$=[2,4] , et $V^{ma}{}_1$=$V^{mi}{}_1$=$V^{ma}{}_2$=$V^{mi}{}_2$=0.5 .

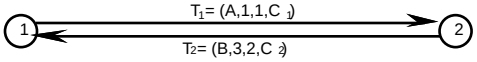
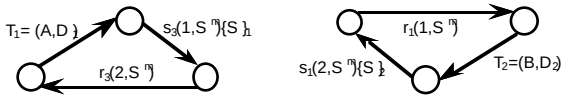



**Figure 10**. *Spécification de service avec contraintes de temps*

Le calcul des contraintes temporelles pour les entités de protocoles, en utilisant les équations [11], [16], [14] et [15], nous donne : $S_1 = [S^{mi}_1, S^{ma}_1]=[1,2]$, $S_2=[S^{mi}_2, S^{ma}_2]=[1.5,3]$, $X_{1,2}= T_{1,2}-M_{1,3}=[1,4]$, $X_{2,1}=T_{2,1}-M_{3,1}=[3,6]$. Les spécifications de protocoles avec contraintes de temps dérivées sont représentées sur la figure 11, avec $D_1=\{X_{2,1}\}$ et $D_2=\{X_{1,2}\}$.

Si l'entité $EP_1$ reçoit le message de l'entité $EP_3$ avec le paramètre $S^m$ (par $r_3(2,S^m)$ ) alors l'intervalle de contrainte de temps sur l'exécution de $T_1$ est $R_{2,1}=[R^{mi}_{2,1}, R^{ma}_{2,1}]$, avec $R^{ma}_{2,1}=X^{ma}_{2,1}-S^m$, et $R^{mi}_{2,1}=\sup(X^{mi}_{2,1}-S^m, 0)$. Si par exemple $S^m = 2$ alors $EP_1$ calcule dynamiquement la contrainte $R_{2,1} = [1,4]$. De même, si $EP_3$ reçoit le message de $EP_1$ avec le paramètre $S^m$ (par $r_1(1, S^m)$) alors l'intervalle de contrainte de temps pour l'exécution de $T_2$ est $R_{1,2}=[R^{mi}_{1,2}, R^{ma}_{1,2}]$, avec $R^{ma}_{1,2} = X^{ma}_{1,2}- S^m$, et $R^{mi}_{1,2} = \sup(X^{mi}_{1,2}-S^m, 0)$.
Si par exemple $S^m = 1.5$ alors $EP_3$ calcule dynamiquement $R_{1,2}=[0, 2.5]$ .




**11.a.** *SP_PT1* **11.b.** *SP_PT3*

**Figure 11** *Spécifications de protocoles avec contraintes de temps*

### *6.3. Second cas dynamique*

Nous reprenons le même exemple de la figure 10 avec les mêmes contraintes $T_{2,1}$, $T_{1,2}$, $M_{1,3}$ et $M_{3,1}$, et les mêmes paramètres $V^{ma}_1$, $V^{mi}_1$, $V^{ma}_2$ et $V^{mi}_2$ que dans le premier cas dynamique. Nous obtenons les mêmes résultats pour $S_1$ et $S_2$. Les spécifications de protocoles sont celles représentées sur la figure 11 si nous remplaçons les $r_a(p, S^m)$ par $r_a(p,S^m, M^m)$, et les $s_a(p, S^m)$ par $s_a(p, S^m, M^m)$ . Avec $D_1 = \{T_{2,1}\}$ et $D_2 = \{T_{1,2}\}$. Si l'entité $EP_1$ reçoit le message de $EP_3$ avec les paramètres $S^m$ et $M^m$ (par $r_3(2,S^m, M^m)$) alors l'intervalle de contrainte de temps pour l'exécution de $T_1$ est $R_{2,1}=[R^{mi}_{2,1}, R^{ma}_{2,1}]$, avec $R^{ma}_{2,1}=T^{ma}_{2,1}-S^m-M^m$, et $R^{mi}_{2,1}=\sup(T^{mi}_{2,1}-S^m- M^m,0)$. Si par exemple $S^m=2$ et $M^m=3$ alors l'entité $EP_1$ calcule dynamiquement la contrainte $R_{2,1}=[0,5]$. De même, si $EP_3$ reçoit le message de $EP_1$ avec les paramètres $S^m$ et $M^m$ (par $r_1(1,S^m, M^m)$ ), alors l'intervalle de contrainte de temps pour l'exécution de $T_1$ est $R_{1,2} = [R^{mi}_{1,2}, R^{ma}_{1,2}]$, avec $R^{ma}_{1,2}= T^{ma}_{1,2}-S^m-M^m$ et $R^{mi}_{1,2}=\sup(T^{mi}_{1,2}-S^m-M^m;0)$. Si par exemple $S^m=1.5$ et $M^m=3$, alors l'entité $EP_3$ calcule dynamiquement la contrainte $R_{1,2}=[0,3.5]$. On remarquera que les contraintes dynamiques $R_{1,2}$ et $R_{2,1}$ pour les entités réceptrices sont moins fortes dans le second cas dynamique que dans le premier.

## 7. Conclusion

Les FSM étant utilisées comme mode de spécification, un algorithme de dérivation des contraintes de temps sur les entités de protocoles à partir des contraintes de

temps sur le service a été développé et sa complexité étudiée. Les calculs sur lesquels il se base sont présentés dans cet article. Les contraintes temporelles peuvent être déterminées statiquement ou dynamiquement. Dans le cas dynamique, nous proposons une méthode d'échanges d'informations temporelles complètes entre les entités de protocoles, qui ne nécessite pas la synchronisation des horloges. Le cas dynamique est intéressant car les entités réceptrices exploitent d'une manière optimale le temps qui leur est réservé afin de satisfaire le service désiré. La méthode de synthèse de protocoles de [Sale90] a été étendue en y intégrant notre algorithme. Mais il faut noter que l'algorithme de génération des contraintes de temps peut être utilisé dans d'autres applications que la dérivation de protocoles. Il peut par exemple être utile, dans le second cas dynamique, pour que des entités réceptrices , qui reçoivent dans les messages une estimation de la durée de séjour dans le medium, rejettent des datagrammes qui ont circulé trop longtemps dans le réseau. Nous pouvons aussi l'utiliser dans des domaines autres que les télécommunications (robotique, ateliers flexibles), où plusieurs systèmes doivent interagir pour accomplir des tâches dans des délais fixés. A partir des contraintes temporelles sur la tâche globale à accomplir, nous générons les contraintes de temps que chacun des systèmes en interaction doit respecter. Mais il faut souligner une limitation: l'absence de concurrence. Les principales améliorations que nous proposons d'apporter ultérieurement sont:

- la prise en compte des tâches concurrentes,
- le traitement des contraintes temporelles entre des événements non consécutifs. Une étude en ce sens est déjà en cours.

**Bibliographie**


[Alur90] R. ALUR, C. COURCOUBETIS et D. DILL, " Model Checking for Real-Time Systems ", Proceedings du 5ème Symposium "Logic in computer Science", Juin 1990, pp.414-425.

[Barr79] W.A. BARRETT et J.D. COUCH, " Compiler Construction: Theory and Practice ", Editeur: Science Research Associates, Inc. 1979.

[Bert91] B. BERTOMIEU et M.DIAZ, " Modeling and Verification of Time Dependant Systems using Petri Nets ", IEEE Transactions on Software engineering, Vol.17(3), Mars 1991, pp.259-273.

[Boch86] G.v. BOCHMANN et R. GOTZHEIN, " Deriving Protocols Specifications from Service Specifications ", Proceedings du Symposium ACM SIGCOM '86, Vermont, USA, 1986, pp.148-156.

[Chu88] P.M. CHU et M.T. LIU, " Synthesizing Protocol Specifications from Service Specifications in FSM Model ", Proceedings IEEE Computer Networking Symposium 1988, pp.173-182.

[Khen89] F. KHENDEK, G.v. BOCHMANN et C. KANT, " New results on deriving protocol specifications from services specifications ", Rapport interne No 692, Département d'Informatique et de Recherche Opérationnelle. Faculté des arts et des sciences, Université de Montréal, Juillet 1989.

[Kant92] C. KANT, T. HIGASHINO et 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, Janvier 1992.

[Kap91] M. KAPUS-KOLAR, " Deriving Protocol Specifications from Service Specifications with Heterogeneous Timing Requirements ", Proceedings IEEE Int. Conf. on Software Engineering for real time systems, Royaume-Uni, 1991.

[KapR91] M. KAPUS-KOLAR et J. RUGELJ, " Deriving Protocol Specifications from Service Specifications with Simple Relative Timing Requirements ", Proceedings ISMM Int. Workshop on parallel computing, Italie, 1991.

[Rama85] C.V. RAMAMOORTHY, S.T. DONG et 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(9), Septembre 1985, pp. 886-908.

[Rico92] N. RICO, G.v. BOCHMANN et O. CHERKAOUI, " Model-Checking for Real-Time Systems Specified in LOTOS ", CAV 1992.

[Sid82] D. P. SIDHU, " Rules for Synthesizing Correct Communication Protocols ", ACM SIGCOM comput. Commun., Rev. Vol.12(1), Janvier 1982, pp.35-51.

[Sale90] K. SALEH et R. PROBERT, " A Service-Based Method for the Synthesis of Communications Protocols ", International Journal of Mini and Microcomputers, Vol.12(3), 1990, pp.97-103.

[Zafi80] ZAFIROPULO, C.H. WEST, H. RUDIN, D.D. COWAN, et D. BRAND, "Towards Analyzing and Synthesizing Protocols ", IEEE Transactions on Communications, Vol.28(4), Avril 1980, pp.651-661.

**Ahmed Khoumsi** est ingénieur de l'École Nationale Supérieure de l'Aéronautique et de l'Espace (Sup'Aéro) à Toulouse (France), option Automatique, en 1984. Il effectua son doctorat en Robotique au Laboratoire d'Automatique et d'Analyse des Systèmes (LAAS) du CNRS dans le groupe Robotique et Intelligence Artificielle (RIA) à Toulouse. Il obtint le diplôme de Doctorat de l'Université Paul Sabatier à Toulouse, en 1988. De 1989 à 1992, il est professeur assistant en Informatique Industrielle et Robotique à l'École Nationale Supérieure d'Électricité et de Mécanique, à Casablanca (Maroc). Depuis Janvier 1993, il est chercheur post-doctoral dans le groupe Protocoles du Département d'Informatique et de Recherche Opérationnelle de l'Université de Montréal. Il travaille actuellement essentiellement sur la synthèse de contrôleurs des systèmes à événements discrets temps-réel.

**Gregor v. Bochmann** a reçu une maîtrise en physique de l'Université de Munich en Allemagne en 1968 et un Ph.D. de l'Université McGill à Montréal au Canada en 1971. Il a travaillé dans les domaines suivants: langages de programmation, théorie des compilateurs, protocoles de communication et ingénierie de logiciels et a publié plusieurs articles dans ces domaines. Il dirige la Chaire de recherche industrielle IDACOM-CRSNG-CCRIT en protocoles de communication au département d'informatique et de recherche opérationnelle de l'Université de Montréal. Il travaille actuellement au développement et à la conception des méthodes de protocoles de communication et de systèmes distribués. Il a été activement impliqué dans la standardisation des techniques de description formelle pour OSI. De 1977 à 1978, il a été professeur invité à l'École Polytechnique Fédérale de Lausanne en Suisse. De 1979 à 1980, il a été professeur invité au Computer Systems Laboratory de l'Université Stanford à Palo Alto en Californie. De 1986 à 1987, il a été chercheur invité pour la compagnie Siemens à Munich en Allemagne.

**Rachida Dssouli** a reçu en 1981 le doctorat de 3ème cycle en informatique à l'université Paul Sabatier à Toulouse en France et le Ph.D. en 1987 à l'université de Montréal au Canada. Elle a été Maître Assistante à l'université Mohamed 1er Oujda au Maroc de 1981 à 1983, et maître de conférences de 1987 à 1989. Elle a été professeure adjointe à l'université de Sherbrooke au Canada de 1989 à 1991. Elle est actuellement professeure adjointe associée à la chaire industrielle au département d'informatique et recherche opérationnelle de l'Université de Montréal. Ses intérêts de recherche sont dans l'ingénierie de protocoles de communication et le génie des logiciels de communication incluant la spécification, test et observation, testabilité, dérivation systématique d'implémentation et évolution des systèmes.