Modélisation au moyen des réseaux de Petri temporisés stochastiques d'une application de contrôle-commande de poste de transformation d'énergie électrique répartie sur le réseau de terrain FIP

Nathalie Bergé

Année 1996

Thèse

Présentée au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS en vue de l'obtention du grade de DOCTEUR de l'UNIVERSITE PAUL SABATIER DE TOULOUSE (Sciences)
SPECIALITE : Informatique Industrielle

par **Nathalie BERGÉ**

MODÉLISATION AU MOYEN DES RÉSEAUX DE PETRI TEMPORISÉS STOCHASTIQUES D'UNE APPLICATION DE CONTRÔLE-COMMANDE DE POSTE DE TRANSFORMATION D'ÉNERGIE ÉLECTRIQUE RÉPARTIE SUR LE RÉSEAU DE TERRAIN FIP

Soutenue le 31 Mai 1996, devant le Jury :

MM.	M. DIAZ	*Président*
	J.P. ELLOY J.P. THOMESSE	*Rapporteurs*
	G. JUANOLE	*Directeur de thèse*
	M. SAMAAN B. SOULAS P. LADET	*Examinateurs*

Rapport LAAS N°96266
Thèse préparée au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
7, Avenue du Colonel Roche, 31077 Toulouse Cedex, FRANCE

Remerciements

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique, dirigé par Monsieur A. Costes que je remercie pour son accueil.

Je remercie Monsieur M. Diaz pour m'avoir accueillie au sein du groupe "Outils et Logiciels pour la Communication".

Je remercie Monsieur G. Juanole pour avoir dirigé cette thèse et pour sa disponibilité tout au long de ce travail.

Je tiens également à remercier tout particulièrement Messieurs B. Soulas et M. Samaan de EDF pour leur conseil et leur soutien permanent tout au long de ce travail. Qu'ils trouvent ici l'expression de ma gratitude et de ma plus sincère sympathie.

Je remercie :

Monsieur M. DIAZ, Directeur de Recherche au CNRS,

Monsieur J.P. ELLOY, Professeur au Laboratoire d'Automatique de Nantes,

Monsieur J.P. THOMESSE, Professeur à l'Ecole Nationale Supérieure d'Electricité et de Mécanique de Nancy,

Monsieur G. JUANOLE, Professeur à l'Université Paul Sabatier de Toulouse,

Monsieur B. SOULAS, Ingénieur à la Direction des Etudes et Recherches de EDF de Moret sur Loing,

Monsieur M. SAMAAN, Ingénieur à la Direction des Etudes et Recherches de EDF de Chatou,

Monsieur P. LADET, Professeur à l'Institut National Polytechnique de Grenoble

qui m'honorent en acceptant de former le jury de cette thèse.

Je remercie Monsieur M. Diaz pour l'honneur qu'il me fait en présidant ce jury, ainsi que Messieurs J.P. Elloy et J.P. Thomesse pour avoir accepté la charge de rapporteur.

Je remercie tous mes collègues et néanmoins amis du groupe OLC pour leur amitié et leur bonne humeur : J.L. Albacète, Y. Atamna, L & R Carmo, L. Gallon, R. Montes, F. Vasques, F. Vernadat...

Je tiens également à remercier mes collègues et amis de EDF pour m'avoir accueillie et fait découvrir leur région : C. Guibout, J. Guillas, R. Hausberger, L. Picci...

Enfin, un grand remerciement à tous mes amis qui m'ont encouragée et soutenue tout au long de ces années de labeur : C. Bayol, J.P. Delacroix, S. Duffour, M. Duros, M. Kâaniche, F & A.C. Luce, P & I Peyre, F. Torres...

Ce travail n'aurait pu être réalisé sans la disponibilité et l'efficacité de tout le personnel du LAAS que je remercie également.

Enfin, je dédie ce mémoire à mes parents et à ma petite soeur, Marylène.

Modélisation au moyen de Réseaux de Petri Temporisés Stochastiques d'une application de contrôle-commande de poste de transformation d'énergie électrique répartie sur le réseau de terrain FIP

Nathalie **BERGÉ**

Soutenue le 31 Mai 1996

Table des matières

Introduction

L'évolution technologique de ces dernières années, notamment dans le domaine des réseaux de communication, a permis la conception de systèmes distribués dans de nombreux secteurs de notre société dont le secteur industriel. En particulier, les réseaux locaux de communication permettent le développement d'applications embarquées, comme dans l'automobile avec le réseau CAN (*Controller Area Network*), et d'applications réparties, comme dans le domaine de la production manufacturière avec le réseau FIP (*Factory Instrumentation Protocol*)...

Ainsi, au sein d'EDF, des études ont été entreprises afin de définir un système distribué pour la surveillance d'un poste de transformation d'énergie électrique HTB/HTA qui est un dispositif de base pour la distribution de l'énergie électrique. Outre la fonction première qui est d'abaisser la Haute Tension à la Moyenne Tension, le rôle d'un poste HTB/HTA est d'assurer le traitement des défauts électriques qui apparaissent sur les lignes électriques afin d'une part, de garantir la sécurité du matériel et des clients et d'autre part, la continuité du service aux clients. L'objectif d'un système distribué pour un poste HTB/HTA est d'augmenter la qualité du service offert.

Les systèmes distribués du secteur industriel sont des systèmes temps réel (les contraintes temporelles sont un élément fondamental de leur fonctionnement) qui ont de grandes exigences en terme de sûreté de fonctionnement (compte tenu de leur grande criticité, un dysfonctionnement peut avoir des conséquences très graves sur l'environnement et les personnes).

Dans ce contexte, il est donc essentiel d'utiliser des méthodologies de structuration et de hiérarchisation, dès les phases de conception d'un système afin d'en maîtriser la complexité, et également, d'utiliser des techniques de description formelle afin d'analyser le comportement futur du système concerné pour en vérifier les propriétés attendues et évaluer les principales performances.

Les techniques formelles utilisées dans le cadre des systèmes temps réel doivent en particulier, être à même de pouvoir exprimer et analyser les contraintes temporelles. Le modèle "Réseaux de Petri Temporisés Stochastiques", développé au sein du groupe OLC (Outils et Logiciels pour la Communication) du LAAS-CNRS, est un modèle particulièrement adapté à l'étude des systèmes distribués temps réel. En effet, il permet d'une part, d'exprimer les principaux mécanismes de communication (parallélisme, synchronisation) et les contraintes temporelles et d'autre part, il offre des possibilités d'analyses qualitative (logique des mécanismes) et quantitative (performances fonctionnelles et de sûreté de fonctionnement).

C'est dans ce double contexte, système réparti de commande et de surveillance d'un poste HTB/HTA d'EDF et modèle formel "Réseaux de Petri Temporisés Stochastiques" que se situe notre travail de thèse ; celle-ci s'est déroulée dans le cadre d'une convention CIFRE entre le groupe ACSAR (Analyse Comportementale des Systèmes et d'Automatismes de Réseaux) de la DER (Direction des Études et Recherches) d'EDF et le groupe OLC (Outils et Logiciels pour la Communication) du LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes) du CNRS.

Ce travail est présenté dans ce mémoire qui comprend cinq chapitres.

Le premier chapitre situe le contexte de l'étude effectuée à savoir, d'une part, les systèmes de contrôle-commande et leurs caractéristiques, et d'autre part, les réseaux de terrain et leurs caractéristiques.

Le deuxième chapitre développe une démarche méthodologique pour permettre de représenter formellement la complexité fonctionnelle des systèmes que ce soit, au niveau applicatif ou au niveau communication. Le cadre formel de cette méthodologie est basé sur le modèle "Réseaux de Petri Temporisés Stochastiques".

Le troisième chapitre a pour objectif de présenter le système sur lequel la méthodologie développée au deuxième chapitre sera appliquée c'est-à-dire, le système de contrôle-commande distribué du poste de transformation d'énergie électrique HTB/HTA d'EDF et le réseau de terrain FIP qui a été choisi par EDF pour cette étude. Plus précisément, sont présentés d'une part, la problématique du poste HTB/HTA afin d'identifier les caractéristiques de l'application distribuée à concevoir et d'autre part, les services et mécanismes du réseau FIP qui seront utilisés.

Les quatrième et cinquième chapitres présentent l'application de la méthodologie développée au chapitre 2 respectivement, au réseau FIP (couche application) et au système de contrôle commande du poste HTB/HTA.
Le but du chapitre 4 est l'étude des mécanismes de validité temporelle synchrones associés aux services périodiques du réseau FIP. Le but du chapitre 5 est l'étude de l'influence du réseau de terrain FIP sur les fonctionnalités et les performances du système de contrôle-commande réparti du poste HTB/HTA pour le traitement des défauts.

Chapitre 1

Contexte de l'étude

1 Introduction

L'objectif de ce chapitre est de situer notre travail à la fois au niveau du domaine des systèmes considérés et au niveau du formalisme retenu pour permettre une spécification formelle, une analyse et une évaluation de ces systèmes.

Ce chapitre comprend trois parties :

- les première et deuxième parties concernent des présentations générales respectivement sur les systèmes de contrôle–commande en milieu industriel et sur les réseaux de communication en site industriel appelés réseaux de terrain ;
- la troisième partie concerne la présentation de techniques de description formelle et plus particulièrement du modèle formel "Réseaux de Petri Temporisés Stochastiques" à la fois en termes de pouvoir d'expression et de pouvoir d'analyse.

2 Les systèmes de contrôle–commande en milieu industriel

2.1 Présentation générale

Un système de contrôle–commande en milieu industriel est un système informatique qui permet la réalisation d'applications de contrôle–commande temps réel c'est-à-dire, dont le fonctionnement est assujetti à l'évolution dynamique de l'état de l'environnement qui lui est connecté (son procédé) et dont il doit contrôler le comportement [Ell88]. Un système de contrôle–commande est composé d'un ensemble de processus d'application qui représentent les fonctions nécessaires à la prise en compte de l'état du procédé et à sa commande.

L'interface entre le procédé et le système de contrôle–commande est constituée par un ensemble de capteurs, qui fournissent au système de contrôle–commande les images du procédé, et d'actionneurs, qui fournissent au procédé les commandes du système de contrôle–commande.

L'exactitude d'une commande est conditionnée par deux attributs : la justesse de sa valeur et la justesse de sa date d'application au procédé. Ce deuxième attribut, c'est-à-dire le respect des contraintes temporelles, est un aspect fondamental et spécifique aux systèmes de contrôle–commande temps réel.

Un processus peut être activé à intervalles réguliers (processus périodique) ou de manière aléatoire (processus apériodique ou sporadique). Un processus peut ne pas avoir le droit de violer une échéance temporelle sous peine de mettre en péril le procédé contrôlé (processus à contrainte stricte) ou bien il peut, de temps en temps, manquer une échéance sans que le procédé en soit affecté (processus à contraintes relatives).

Un système de contrôle–commande en milieu industriel est, dans le cas le plus général, un système géographiquement distribué ; ceci résulte de la répartition géographique (imposée ou non) des différents équipements du procédé et/ou de la distribution des éléments de contrôle–commande (imposée ou voulue). Les contraintes géographiques sont imposées, soit par les aspects matériels des équipements, i.e. procédés distants (exemple des différentes zones de transferts, d'usinages, de stockages,... dans un atelier de production), soit par des contraintes de protection des ressources informatiques qui doivent être éloignées du procédé en environnement perturbé (exemple des calculateurs ou systèmes de contrôle-commande dans une application de régulation de température d'un four).

Un tel système permet de réaliser des applications dites "applications de contrôle–commande temps réel distribuées".

Les communications, d'une part, entre les capteurs/actionneurs et les processus de contrôle–commande qui leur sont associés et d'autre part, entre les différents processus de contrôle–commande, représentent une activité fondamentale pour le bon fonctionnement d'une application de contrôle–commande temps réel distribuée. En effet, tous ces éléments requièrent un grand potentiel de communication pour échanger des informations (consignes, états, alarmes,...) qui doivent respecter un ensemble de contraintes temporelles imposées par les évolutions dynamiques du procédé. Dans ce contexte, on parle de communications temps réel. Notons que, comme on a des processus périodiques et des processus apériodiques, l'exécution de ces différents processus va induire des communications périodiques et apériodiques.

Dans le cadre de notre travail, nous considérons une application de contrôle–commande temps réel distribuée pour un poste de transformation d'énergie électrique de EDF (poste HTB/HTA). Cette application est située dans un environnement, évidemment perturbé, en milieu extérieur soumis aux intempéries et aux perturbations électromagnétiques ;elle sera décrite au chapitre 3.

2.2 Sur les communications

Actuellement encore, un grand nombre de systèmes temps réel géographiquement distribués utilisent des liaisons point à point entre les différents sites (équipements). La logique de

communication entre les sites est alors très simple : il n'y a pas de partage des ressources de communication et donc, pas de problème de gestion inhérente à ce partage. En outre, le temps de communication des informations entre les équipements (sites) est négligeable et seuls les temps de traitement des informations doivent être considérés pour vérifier la satisfaction des contraintes temporelles. Cependant, la modularité de la technique des liaisons point à point est faible ; cette technique n'offre pas une grande flexibilité pour des évolutions futures du système.

L'apparition des réseaux locaux industriels a eu pour avantage de permettre l'interconnexion des équipements d'une application par un câble de communication unique constituant une ressource partagée entre tous les équipements et garantissant ainsi la facilité d'ouverture. Cependant, la nécessité de multiplexer les informations des différents sites sur la ressource de communication complexifie son utilisation et augmente donc les temps d'échanges des informations dont il est, maintenant, impératif de s'assurer qu'ils sont bornés de manière à ce que les contraintes temporelles des applications soient satisfaites. Dans le cadre des applications de contrôle–commande temps réel, les réseaux locaux utilisés entre les niveaux 0 et 1 de l'architecture CIM [She94] sont appelés des Réseaux de Terrain.

Les capteurs et actionneurs étant proches du procédé, ils peuvent donc être situés dans des environnements très perturbés (chaleur, intempéries,...). De plus, bien souvent les échanges entre capteurs/actionneurs et leurs processus de contrôle–commande associés doivent être très rapides ; l'accès à une ressource commune (le réseau de communication) peut donc affecter de manière non négligeable cette contrainte de rapidité. En outre, si les capteurs/actionneurs étaient connectés au réseau de communication, leur environnement perturbé pourrait également affecter les communications sur le réseau.

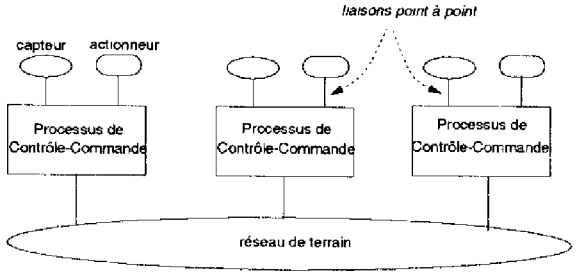

Figure 1: systèmes de contrôle-commande répartis sur un réseau de terrain

Dans le contexte de l'application que nous considérons (contrôle-commande de poste HTB/HTA), il est préférable d'adopter le plan d'interconnexion de la figure 1 où :

- **les capteurs et actionneurs sont reliés par des liaisons point à point à leur processus de contrôle–commande.** En effet, étant situés en milieu extérieur et donc soumis aux intempéries et aux perturbations électromagnétiques, ils doivent être éloignés du système de contrôle-commande ;

- **les processus de contrôle–commande sont connectés entre eux par le réseau de terrain.** Ils peuvent ainsi être éloignés du procédé et situés en site protégé.

2.3 Sur les architectures pour les systèmes temps réel

Il existe deux grandes classes d'architectures pour des systèmes temps réel, dit aussi réactifs [KDK+89] :

- les architectures *Event-Triggered Systems* (ET), c'est-à-dire guidées par les évènements, où les activités du système de contrôle sont initialisées par l'occurrence d'évènements significatifs (changements d'état) de l'environnement ou du système informatique de structure sous-jacent ;

- les architectures *Time-Triggered Systems* (TT), c'est-à-dire guidées par le temps, où les activités du système de contrôle sont initialisées à des instants périodiques prédéterminés dans la base de temps globale synchronisée de manière à observer périodiquement les états de l'environnement.

Dans les architectures ET, les changements d'états du procédé, vus sur un site, doivent être signalés rapidement aux autres sites répartis par la transmission d'un message urgent. Ceci nécessite de prendre une décision d'ordonnancement en temps réel pour réaliser le transfert et il n'est pas évident que les contraintes temporelles de ce type de message soient satisfaites : cela dépend de la charge du réseau, de l'état des files d'attente contenant les messages à transférer, etc.

Or, la rapide diffusion des états observés du procédé temps réel entre tous les sites d'un système de contrôle réparti est une affaire primordiale. Celle-ci peut être atteinte dans les architectures TT par la diffusion périodique des informations relatives aux états observés du procédé dans chaque site (avec des périodes qui égalisent la dynamique du procédé commandé) afin d'avoir une vue *en temps réel* des états du procédé [Kop]. Ainsi, dans une architecture TT (par rapport à une architecture ET), il est possible de prévoir les échanges nécessaires et donc, il est généralement plus facile de résoudre les problèmes fondamentaux de communication temps réel (contrôle de flux, ordonnancement, satisfaction des contraintes temporelles, détection des erreurs en temps réel...). Cette facilité est due à la régularité imposée à ce type de systèmes par la périodicité des mécanismes mis en œuvre et la caractérisation statique (prédéterminée à l'avance) de leurs besoins en échanges et de leur contrainte temporelle [Kop]. Le contrôle de flux est réalisé de manière implicite par le fait que les échanges sont prédéterminés et donc le nombre de messages échangés par unité de temps est prévisible et pratiquement constant. L'ordonnancement des échanges est facilité par la connaissance globale au préalable de l'ensemble des échanges nécessaires et de leurs contraintes temps réel. De plus, des intervalles de temps peuvent être laissés pour la réalisation des transferts de messages apériodiques (urgents ou non). Les instants de transmission des messages sont alors connus à l'avance à priori par tous les partenaires de la communication. La détection des erreurs (retard ou perte de messages) peut ainsi être effectuée en

temps réel (en supposant que si un message n'est pas reçu à un instant précis, c'est une erreur). Le stockage des valeurs des états du procédé dans le système de communication est à la charge de ce dernier. Il est généralement effectué dans des buffers placés dans les interfaces de communication : une écriture, dans un buffer, d'une donnée produite par le procédé écrase la valeur précédente et une lecture dans le buffer, par un site consommateur, d'une donnée reçue périodiquement (relative à l'état d'un élément du procédé) ne détruit pas la valeur contenue dans ce buffer.

En conclusion, il nous apparaît que les architectures TT (qui présupposent évidemment une connaissance exhaustive du comportement du système à concevoir) présentent, par rapport aux architectures ET, les avantages suivants :

- **plus grande facilité de mise en œuvre** (contrôle de flux, ordonnancement statique...),
- **plus grande prédiction du comportement.**

3 Les réseaux de terrain

3.1 Architecture

Les réseaux de terrain représentent, par rapport aux réseaux de communication classiques (architecture ISO de l'OSI [Zim80]) une modification (réduction) en terme d'architecture, précisément dans le but de diminuer les délais de communication des échanges d'informations entre les éléments du procédé pour lequel il est important de surveiller et contrôler les évolutions en temps réel (pour des raisons de sécurité du procédé et de l'environnement).

L'architecture d'un réseau de terrain est donc une architecture réduite aux trois couches essentielles de l'architecture ISO en sept couches (figure 2) : application – liaison de données – physique.

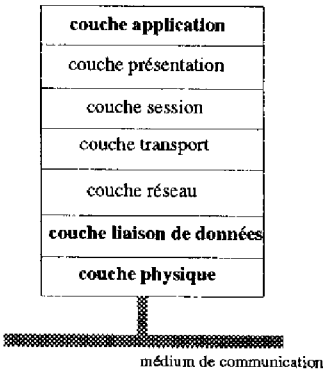

Figure 2: architecture OSI

Dans le contexte temps critique des applications industrielles, il faut s'assurer que les échanges se font dans des intervalles de temps déterminés (bornés et généralement connus) de telle façon que les contraintes temporelles strictes des processus industriels soient satisfaites. Les principales caractéristiques que doivent satisfaire les réseaux de terrain sont donc : temps de transfert déterminé, contrôle du flux, faible longueur des trames, capacité à traiter/gérer des évènements particuliers (type synchronisation), génération d'un trafic périodique (ou cyclique) en plus du trafic apériodique (ou acyclique) et mise en œuvre de mécanismes particuliers permettant la vérification des contraintes temporelles. La satisfaction des contraintes temporelles ne peut être assurée que par un bon contrôle de l'accès au médium (MAC).

3.2 Les principaux types de contrôle d'accès au médium

Les principaux types de contrôle d'accès au médium sur les réseaux de terrain les plus connus (FIP [AFN90a], Profibus [Deu90], SP50 [ISA]) sont : soit distribués sur chaque (ou plusieurs) station(s) ; soit centralisés sur une station unique (nous ne parlerons pas ici des protocoles basés sur CSMA/CD, DCR [IEE92] et CAN [Fre92] [ISO93]).

Contrôle d'accès distribué : lorsque le contrôle de l'accès au médium est distribué, ceci signifie qu'il est successivement attribué aux différentes stations connectées sur le médium :

- soit en faisant circuler un jeton entre les stations (qu'elles conservent pendant un certain temps et se transmettent). Par exemple, dans Profibus, le contrôle de l'accès au médium est donné à une station (appelée station maître) qui devient responsable des échanges pendant son temps de possession du jeton. Le temps d'allocation du jeton à une station peut varier en fonction de la station : seul le temps global de rotation souhaité est précisé.

- soit en allouant l'accès à chaque station pendant un temps fixe et prédéterminé. Par exemple, dans le protocole TTP [KDK+89], chaque station accède au médium pendant un intervalle de temps prédéterminé ; ceci est basé sur l'hypothèse d'un partage du temps global entre toutes les stations. Chaque station a la connaissance de ce temps global (horloge globale synchronisée par des algorithmes de synchronisation) et de l'intervalle de temps qui lui est réservé. Il s'agit alors d'un multiplexage temporel implicite de l'accès au médium (par connaissance au préalable).

Contrôle d'accès centralisé : l'accès au médium est centralisé lorsqu'une station unique particulière (appelée gestionnaire) est responsable de la gestion des droits d'accès au médium. Elle attribue le droit de parole successivement aux différentes stations :

- soit en envoyant un message à chaque station pour leur donner le droit de parler à un instant précis. Par exemple, les réseaux de terrain FIP et IEC 65C [IEC] possèdent un gestionnaire, appelé respectivement "Arbitre de Bus" (BA) et "Link Access Scheduler" (LAS), qui envoie des messages aux différentes stations leur permettant alors d'émettre.

- soit en jouant le rôle d'une horloge globale centralisée qui alloue le droit d'accès aux stations pendant des intervalles de temps spécifiques. On parle de multiplexage temporel. Par exemple, le protocole TDMA (Time Division Multiplexing Access) [Rol95] où l'entité centralisée génère des trames qui sont découpées en sous trames et chacune d'elles est spécifiquement allouée à une station distincte.

3.3 Le type d'accès considéré

Le contrôle d'accès centralisé paraît le mieux adapté, dans une optique d'architecture guidée par le temps, à la gestion d'échanges d'information temps réel entre procédés industriels : d'une part, parce que le contrôle de l'accès est géré par une entité unique qu'il peut être nécessaire d'isoler/éloigner des équipements de mesures tels que capteurs et actionneurs dans un environnement perturbé (haute température, forts champs électromagnétiques...) et d'autre part, parce qu'il est possible de prévoir et d'ordonner de façon déterministe les échanges connus résolvant ainsi, le problème de contrôle du flux, du respect des contraintes temporelles strictes imposées par les besoins des procédés,... En ce qui concerne le gestionnaire, nous considérons que seules les couches physique et liaison sont nécessaires pour l'étude de l'ordonnancement et la gestion des accès au médium de communication.

Dans le cadre de notre travail, nous considérons (figure 3) un réseau de terrain à architecture réduite et accès centralisé (FIP) sur lequel est connectée l'application de contrôle-commande via les interfaces de communication. Les caractéristiques principales du réseau de terrain FIP seront présentées au chapitre 3.

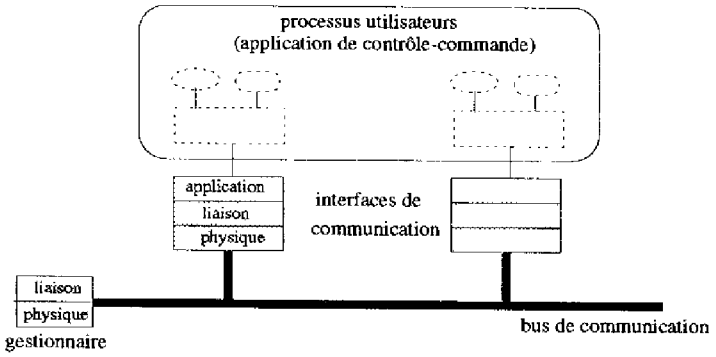

Figure 3: processus utilisateurs répartis sur un réseau de terrain

4 Les techniques de description formelle

4.1 Généralités

Les systèmes distribués sur des réseaux de communication présentent de grandes difficultés pour leur conception du fait de la distribution des équipements et des interactions nécessaires à la réalisation d'une application répartie. La conception de tels systèmes nécessite l'utilisation d'une part, de méthodologies appropriées à leur développement et d'autre part, de techniques de description formelle, préconisées dès les phases de conception pour la vérification et la validation du système dans le but d'éliminer des erreurs de conception avant la phase d'implantation et ainsi minimiser les coûts de réalisation.

Généralement, il est tout d'abord nécessaire de spécifier l'architecture du système (afin de déterminer ses éléments et leurs interactions), puis de décrire le comportement dynamique des différents éléments du système de façon formelle. A ce niveau, le choix des modèles formels utilisés dépend des analyses souhaitées et des résultats attendus. Soulignons ici que deux types de techniques formelles cœxistent, les modèles et les langages, et que deux types d'analyses sont nécessaires, les analyses qualitatives (qui portent sur le comportement logique du système) et les analyses quantitatives (qui portent sur les performances du système).

Dans le chapitre 2, nous présentons la démarche méthodologique de structuration et de modélisation des comportements dynamiques que nous appliquerons à : l'étude du réseau de terrain FIP dans le chapitre 4 et à l'étude du système de contrôle-commande de poste de transformation d'énergie électrique réparti sur FIP dans le chapitre 5.

4.2 Les approches modèles et langages

Les approches modèles sont généralement basées sur les réseaux de Petri ou machines à états... ; les approches langages reposent sur les techniques de description formelle telles que ESTELLE ou LOTOS...

Les approches modèles : leur intérêt est de permettre une représentation (simple) des mécanismes principaux (de base) des systèmes tels que : le parallélisme, la synchronisation, la compétition et les échanges de messages. Cependant, une représentation complète des spécifications n'est pas permise de par les structures utilisées et la complexité croissante des modèles si l'on considère des systèmes de grandes dimensions. Par contre, une analyse exhaustive (vérification) des mécanismes représentés est possible.

Les modèles particulièrement utilisés sont basés sur les Réseaux de Petri (RdP) [Rei85] [Bra83] et proposent : soit des abréviations de haut niveau pour fournir des représentations concises (introduction de couleurs et de variables) avec les RdP colorés [Jen84] ou Prédicats/Transitions [Gen86] Étiquetés [Llo90] ; soit des extensions en introduisant des caractéristiques temporelles avec les RdP temporels [Mer76], temporisés [Ram74] ou stochastiques [Nat80] [Mol81] ou Temporisés Stochastiques (RdPTS) [JA93].

Les approches langages : leur but est de permettre la représentation de la totalité des mécanismes spécifiés (exhaustivité de la représentation). Leur principal avantage est de permettre une description détaillée complète permettant, la plupart du temps, une implantation par la compilation du langage (génération automatique de code). Deux des principaux langages définis pour le domaine des réseaux de communication sont : ESTELLE [ISO] basé sur la notion d'automates communicants et LOTOS [BB87] basé sur la notion d'algèbre de processus (CCS) [Mil80]. Cependant, une analyse exhaustive de la spécification n'est pas souvent envisageable du fait des nombreux mécanismes représentés ; des vérifications partielles peuvent être menées par la simulation ou la génération de séquences de test. On peut encore citer les langages synchrones, utilisés dans le domaine des systèmes distribués réactifs avec ESTEREL [BG92], LUSTRE et SIGNAL [BB91].

Soulignons que les deux approches, modèles et langages, se complètent efficacement en apportant la vérification respectivement des principaux mécanismes et des mécanismes plus orientés implantation (gestion de buffers...).

Dans le cadre de notre travail, nous considérons le modèle "Réseaux de Petri Temporisés Stochastiques" [JA93] qui est un modèle bien adapté pour la spécification et l'analyse des systèmes distribués temps réel et pour la vérification de propriétés qualitatives et quantitatives. Ce modèle sera utilisé dans les chapitres 2, 4 et 5.

4.3 Les Réseaux de Petri Temporisés Stochastiques

Le modèle "Réseaux de Petri Temporisés Stochastiques" (RdPTS) étend le modèle Réseau de Petri (RdP) avec des spécifications temporelles et stochastiques [Ata94] sur les transitions. L'intérêt de ce modèle est d'une part, de permettre la modélisation des mécanismes fondamentaux des systèmes distribués temps réel (tels que le parallélisme, la synchronisation et les caractéristiques temporelles) et d'autre part, de fournir un cadre unique pour permettre des analyses qualitatives (logique des mécanismes) et quantitatives (évaluation des performances fonctionnelles et de sûreté de fonctionnement).

4.3.1 Définition formelle

Un Réseau de Petri Temporisés Stochastiques est défini par un triplet (PN,IO,FO) où :

- PN est le Réseau de Petri (RdP) sous-jacent place/transition avec des arcs inhibiteurs ;
- IO est la fonction intervalle de tir initial. A chaque transition t_i est associé un intervalle de temps $[\theta_{mi}, \theta_{Mi}]$. Nous avons : $0 \leq \theta_{mi} \leq \theta_{Mi} \leq \infty$; θ_{mi} et θ_{Mi} correspondent aux dates de sensibilisation de la transition et sont respectivement : la date de tir au plus tôt et la date de tir au plus tard. Le tir est instantané ;
- FO est la fonction de densité de probabilité de tir. A chaque transition t_i est associée une fonction de densité de probabilité $f_i(x)$ dans son intervalle de tir initial. Nous avons $\int_{\theta_{mi}}^{\theta_{Mi}} f_i(x)dx = 1$.

Une fonction de densité de probabilité peut être : continue (uniforme ou exponentielle), discrète (impulsion de Dirac), ou mixte (uniforme et discrète). Les densités de probabilités discrètes, uniformes et mixtes permettent la représentation des contraintes temporelles.

4.3.2 Analyse

L'analyse du comportement dynamique des Réseaux de Petri Temporisés Stochastiques est effectuée par une analyse d'accessibilité basée sur l'objet "graphe d'états probabilisé". Ce graphe permet à la fois des analyses qualitatives et quantitatives.

4.3.2.1 Le graphe d'états probabilisé

Ce graphe est composé d'états et de transitions entre états.

- Un état est un triplet (M, I, F) où : M est le marquage du réseau, I est l'ensemble des intervalles de tir des transitions sensibilisées par le marquage M et F est l'ensemble des fonctions de densité de probabilité sur les intervalles de tir des transitions sensibilisées.
- Une transition entre deux états est un triplet $< t_i, \theta_i, p_i >$ où t_i est le nom de la transition du réseau de Petri sous-jacent dont le tir provoque le changement d'état, θ_i est le temps moyen de tir (ou temps conditionnel de séjour) dans l'état d'entrée de la transition et p_i est la probabilité de branchement associée. Les valeurs de θ_i et p_i dépendent de l'intervalle de temps, de la fonction de densité de probabilité de la transition t_i et des transitions sensibilisées simultanément (s'il y en a).

Les conditions de tir d'une transition t_i à un instant θ_i avec une probabilité p_i sont :

- t_i est sensibilisée dans le réseau de Petri sous-jacent ;
- θ_i est compris entre la borne minimale de l'intervalle de tir associé à la transition t_i et la plus petite valeur des bornes maximales des intervalles de tir de toutes les transitions sensibilisées ;
- la probabilité de tir p_i, entre la borne minimale de tir de t_i et la plus petite valeur des bornes maximales des intervalles de tir de toutes les transitions sensibilisées, est non nulle.

4.3.2.2 Exploitation du graphe d'états probabilisé

L'exploitation du graphe d'états probabilisé peut être faite à deux niveaux complémentaires.

Premier niveau : on ne considère pas les probabilités p_i et les temps θ_i associés aux arcs de changements d'états du graphe ; on considère uniquement la structure du graphe et la sémantique des actions associées aux arcs (envoi de message, réception de message, évènement interne...). Ainsi, on peut, à l'aide de ce graphe, faire une analyse qualitative du

comportement du système étudié, de façon identique aux analyses faites sur les systèmes de transitions étiquetés classiques ; seulement ici, différence très importante, la structure du graphe obtenu dépend des caractéristiques temporelles associées aux transitions du modèle RdPTS. L'analyse qualitative permet d'analyser deux types de propriétés :

- des propriétés générales telles que bornitude, vivacité, cyclicité... ;
- des propriétés spécifiques qui dépendent de la mission du système modélisé ; ces propriétés sont obtenues à partir d'une interprétation de la sémantique des places et des transitions relativement aux états atteints dans le graphe. On peut, en particulier :
  - identifier des séquences d'évènements et établir des relations entre deux évènements sous la forme de formules de logique temporelle : *il est possible que..., il est inévitable que...* ;
  - obtenir des vues abstraites (ou automates quotients) qui représentent le comportement du système réduit par rapport à un sous-ensemble d'évènements déclarés *observables* (on peut utiliser en particulier les techniques de projection basées sur la relation d'équivalence observationnelle [Mil80]).

Deuxième niveau : on considère maintenant les valeurs des probabilités p_i et des temps θ_i, ce qui permet de faire des analyses quantitatives. Les analyses peuvent être faites en régime transitoire et en régime permanent. L'analyse en régime transitoire concerne, en particulier, l'évaluation des probabilités des différents chemins de séquences d'évènements. Si le graphe d'états probabilisé a une composante terminale fortement connexe, l'analyse en régime permanent peut être effectuée à partir des deux matrices P (matrice des probabilités de transition p_i) et Θ (matrice des temps conditionnels de séjour θ_i) :

- le vecteur des probabilités à l'équilibre γ est obtenu à partir de la matrice P en résolvant l'équation :
$$\gamma^t.P = \gamma^t \; avec \sum_i \gamma_i = 1$$
- les temps moyens inconditionnels de séjour (η_i) (dans l'état i) sont obtenus en combinant les matrices P et Θ tel que :
$$\eta_i = \frac{\sum_j p_j \theta_j}{\sum_{j,j \neq i} p_j}$$
où j est relatif à l'ensemble des arcs sortants de l'état i.
- les probabilités en régime permanent π_i de chaque état i sont calculées à partir du vecteur des probabilités à l'équilibre γ_i et des temps inconditionnels de séjour η_i relatif à l'état i :
$$\pi_i = \frac{\eta_i \gamma_i}{\sum_j \eta_j \gamma_j}$$
où j est l'ensemble des indices des états du régime permanent.

4.3.2.3 Vues abstraites quantitatives

Définition :

Une vue abstraite quantitative d'un graphe d'états probabilisé est un graphe réduit obtenu, à partir du graphe d'états probabilisé, en faisant une projection sur un sous-ensemble d'états (appelés états observables) de ce graphe d'états probabilisé de la manière suivante : les probabilités de branchement et les temps moyens de transition entre les états du graphe réduit (états observables) sont les probabilités et les temps moyens de premier passage entre ces états dans le graphe d'états probabilisé, compte tenu des cheminements possibles seulement à travers les états non observables [AJ95].

Les états définis comme états observables sont des états pertinents pour l'analyse du point de vue du concepteur.

Une vue abstraite quantitative est une chaîne immergée réduite, à partir de laquelle des indices de performances, en régime transitoire et/ou en régime permanent (si on a au moins une composante fortement connexe) relativement au monde des évènements pertinents pour un utilisateur, peuvent être calculés avec une complexité et un coût de calcul réduit (pour rapport au calcul qu'il faut faire avec le graphe d'états probabilisé). Des exemples d'utilisation des vues abstraites quantitatives sont donnés dans [Car94] et [Ata94].

Nous présentons ici la méthodologie de calcul d'un indice de performances en sûreté de fonctionnement que nous utiliserons plus particulièrement dans ce travail au chapitre 4 : c'est le temps moyen d'occurrence d'une erreur (Mean Time to First Failure, noté MTFF) dans le déroulement d'une action.

Calcul du MTFF :

Considérons un graphe d'états probabilisé et appelons $E = \{E_i\}$, l'ensemble des n états qui représentent le début d'une action et $Ee = \{E_j\}$, l'ensemble des m états qui traduit un résultat incorrect de cette action (c'est donc une erreur).

Le calcul du MTFF comprend trois étapes :

1. tout d'abord, on considère la vue abstraite relative aux seuls états E_i, c'est-à-dire on visualise le monde de l'occurrence du début de l'action ; à partir de cette vue abstraite, on obtient la probabilité à l'équilibre γ_i de ces états E_i (cette probabilité représente la proportion de transition dans chaque état E_i) ;

2. ensuite, on considère successivement toutes les vues abstraites obtenues en faisant une projection :
   - sur un état E_i ;
   - l'ensemble des m états d'erreur E_j.

   On obtient le MTFF relatif à l'état E_i (noté MTFF_i) de la manière suivante :

$$MTFF_i = \sum_{j=1}^{m} F_{i,j}.T_{i,j}$$

où $F_{i,j}$ et $T_{i,j}$ sont respectivement la probabilité de premier passage et le temps de premier passage de l'état E_i à l'état d'erreur E_j ;

3. enfin, on obtient le MTFF relatif à l'action analysée :

$$MTFF = \sum_{i=1}^{n} \gamma_i . MTFF_i$$

4.3.3 L'outil RPTS

L'outil RPTS permet d'une part, l'édition du modèle RdPTS et d'autre part, la construction du graphe d'états probabilisé, ainsi que l'obtention de vues abstraites quantitatives, à partir du graphe d'états probabilisé, pour effectuer des analyses qualitatives et quantitatives. Il a été développé en langage C, au LAAS [Ata93] [Ata94], et tourne sous SunOS 4 et Solaris 2.

Nous utilisons cet outil pour les études effectuées aux chapitres 4 et 5.

5 Conclusion

Dans le cadre de ce chapitre, nous avons voulu mettre l'accent d'une part, sur la problématique des systèmes de contrôle-commande en environnement industriel et des réseaux de terrain utilisés dans un tel contexte et d'autre part, sur l'intérêt du modèle "Réseaux de Petri Temporisés Stochastiques" pour la modélisation et l'analyse de tels systèmes.

La problématique des systèmes de contrôle-commande en milieu industriel est la problématique des systèmes réactifs distribués, c'est-à-dire elle s'exprime en termes de contraintes temporelles sur les activités de l'application qui induisent elles-mêmes des contraintes temporelles sur les communications entre les éléments de l'application. La question se pose en particulier, sur le choix d'une architecture (guidée par les évènements ou guidée par le temps). Nous avons indiqué tout l'intérêt, dans un milieu industriel, des architectures guidées par le temps (en supposant que l'on a une connaissance quasi-exhaustive du comportement) qui permettent une plus grande facilité de mise en œuvre et de prédictabilité comportementale.

La problématique du réseau de terrain est de pouvoir transférer dans des délais bornés des informations. Le niveau MAC (*Medium Access Control*) est donc le niveau critique et nous avons indiqué tout l'intérêt, dans le cadre d'une architecture guidée par le temps, d'un contrôle d'accès centralisé (notion de gestionnaire de bus).

Les Réseaux de Petri Temporisés Stochastiques ont un pouvoir d'expression bien adapté aux exigences des architectures de systèmes distribués temps réel (parallélisme, synchronisation, contraintes temporelles) et offrent des possibilités d'analyses qualitatives (vérification de propriétés) et quantitatives (évaluation de performances fonctionnelles et de sûreté de fonctionnement).

Chapitre 2

Méthodologie de modélisation d'un système de contrôle–commande temps réel réparti sur un réseau de terrain à contrôle d'accès centralisé

1 Introduction

Ce chapitre présente la démarche méthodologique générale proposée pour la structuration, la modélisation et l'analyse d'un système complexe de contrôle-commande temps réel réparti sur un réseau de terrain à contrôle d'accès centralisé. L'intérêt de la méthodologie est d'aider le concepteur dans les phases de développement du système en lui indiquant les différentes étapes de conception à suivre afin de réaliser la modélisation et la vérification du système étudié au moyen des Réseaux de Petri Temporisés Stochastiques (RdPTS).

Ce chapitre comprend trois parties :

- la première partie présente l'activité de décomposition structurée qui permet d'obtenir une représentation modulaire fonctionnelle, à la fois pour un système de contrôle-commande réparti et pour un réseau de terrain à contrôle centralisé ;
- la seconde partie présente l'activité de modélisation des comportements dynamiques temporels en RdPTS, basée sur la modularité et sur la composition de modules ;
- la troisième partie présente les étapes de l'analyse des modèles, tirant partie de la modularité, pour la vérification du comportement et des contraintes temporelles des systèmes.

2 Structuration

2.1 Principes généraux

2.1.1 Sur les méthodologies de conception structurées

Une méthodologie est une suite ordonnée (ensemble de démarches successives ou cheminement d'étapes) de méthodes à suivre pour aboutir à l'étude complète d'un système. Une méthode est une technique de résolution d'un problème donné au moyen d'outils (par exemple, techniques de description formelle) caractérisées par des règles et une sémantique (auxquelles des logiciels peuvent être associés).

Différentes méthodologies de conception ont été proposées ces dernières années dans différents domaines tels que : la micro-électronique [CFT86] [Cal90], le génie logiciel [BPMM95] [BDT92] et les systèmes de production [BG91] [Com91] [Cru91] [ACG92]. Ces méthodologies proposent des démarches spécifiques à suivre pour chaque type de systèmes. De façon générale, l'ensemble des travaux montre les besoins de structuration et de décomposition lors des phases de conception.

Deux approches distinctes cœxistent : les approches fonctionnelles qui sont basées sur une représentation du système relativement à ses fonctionnalités (fonctions ou activités attendues) et qui nous concernent plus particulièrement ; les approches orientées objets (plus destinées à la programmation [Bai89] [Boo86]).

Compte tenu du contexte de ces méthodologies, elles ne prennent généralement en compte ni les aspects de répartition liés à l'utilisation de réseaux de communication, ni les aspects temporels liés aux caractéristiques temps réel des systèmes et aux besoins d'études de mise en œuvre sur un réseau ; ceci fera, plus particulièrement, l'objet de la complémentarité de notre étude.

2.1.2 Etapes de structuration

Les trois étapes essentielles de structuration des méthodologies de conception basées sur les approches fonctionnelles comprennent la définition : de l'architecture globale, de l'architecture détaillée et des échanges.

1. **L'architecture globale** : elle est définie par la décomposition systémique de l'application en sous-systèmes distincts ; ceci permet également de préciser quel est l'environnement contrôlé par le système.

2. **L'architecture détaillée** : elle est définie par la décomposition fonctionnelle en blocs fonctionnels de chaque sous-système de l'architecture globale. Chaque bloc fonctionnel peut ensuite être raffiné en un ensemble de sous-blocs fonctionnels de niveau inférieur... jusqu'à obtenir la décomposition fonctionnelle la plus détaillée en blocs ou modules élémentaires. Le nombre de niveaux dépend de la complexité du système et du niveau de description que l'on veut atteindre (laissé au choix du concepteur).

3. **Les échanges** : ils doivent être définis progressivement, à chaque niveau de décomposition de l'architecture (globale et détaillée). Ils sont constitués par l'ensemble des relations (échanges de messages, de données, de contrôle...) entre les différents éléments de l'architecture (sous-systèmes et blocs fonctionnels).

2.1.3 Cadre d'application de ces principes

Dans le cadre de ce travail, nous présentons l'application de ces trois étapes pour la structuration de deux types de systèmes distincts :

- d'une part, un système de contrôle–commande réparti,
- d'autre part, un réseau de terrain à contrôle centralisé.

En ce qui concerne les échanges, nous les définissons dans ce chapitre après les présentations de chaque architecture détaillée.

2.2 Contrôle-commande temps réel réparti

2.2.1 L'architecture globale

L'architecture globale correspond à la décomposition de premier niveau du système global. Elle est définie à partir de la décomposition systémique du système de contrôle-commande global réparti en sous-systèmes distincts de contrôle-commande. L'environnement du système global est défini par l'ensemble des procédés contrôlés et commandés par chaque sous-système de contrôle-commande. Cette décomposition systémique est inhérente aux contraintes topologiques (géographique, matérielle et/ou de protection) liées au procédé.

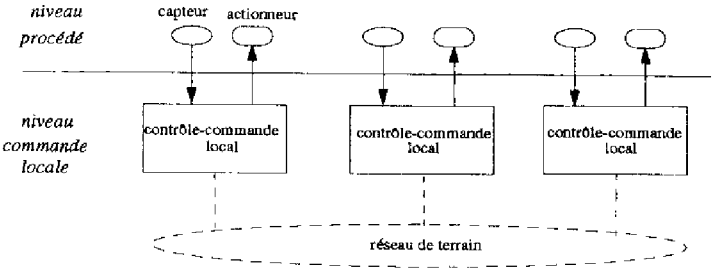

Figure 1: application de contrôle–commande répartie

L'architecture globale retenue, qui est représentée sur la figure 1, est l'architecture déjà considérée au chapitre 1. Elle comprend :

- le système de contrôle-commande réparti qui est composé par l'ensemble des sous-systèmes de contrôle–commande associés aux procédés locaux (appelés sous-systèmes de contrôle–commande locaux) ;

- l'environnement qui est composé par les différents procédés locaux (capteurs et actionneurs) répartis et le réseau de terrain qui constitue le moyen de réalisation des communications.

2.2.2 L'architecture détaillée d'un sous-système de contrôle-commande local

2.2.2.1 Les fonctions

Chaque sous-système de contrôle-commande local peut assurer les fonctions essentielles [Com91] : de *commande* du procédé local et/ou distant ; de *surveillance* (centralisée ou répartie) de l'ensemble des processus distants (à travers les sous-systèmes de contrôle-commande) et du procédé local.

La fonction de *commande locale* agit directement (boucle de contrôle-commande directe) sur le procédé contrôlé et commandé localement, à partir de l'observation des états du procédé local (via l'ensemble des données issues des capteurs et actionneurs locaux) et l'ensemble des états des procédés distants (via leur sous-système de contrôle-commande).

La fonction de *surveillance* assure le contrôle de l'ensemble des procédés en cas de défaillances. Elle comprend les fonctions : de *détection* des dysfonctionnements des procédés par l'analyse des informations (comptes-rendus) issues des procédés locaux et/ou des sous-systèmes distants ; de *diagnostic* (identification de la cause des dysfonctionnements) ; de *décision* (détermination des solutions de reprise à mettre en œuvre pour traiter les dysfonctionnements). L'application des procédures de reprise est effectuée par la fonction de commande dans un souci de cohérence.

La mise en œuvre des fonctions de la surveillance (détection, diagnostic et décision) peut être réalisée de deux manières : soit intégrée à la commande, soit séparée de la commande.

Une *solution intégrée* consiste à prendre en compte les occurrences d'anomalies et leur traitement au niveau de chaque sous-système de commande. Cette solution peut être envisagée lorsque certaines défaillances et leurs procédures de reprise sont connues à l'avance. Ainsi, on peut réagir en temps réel aux défaillances connues du procédé en appliquant les procédures de reprise prévues.

Une *solution séparée* est nécessaire lorsque les types de défaillances ne sont pas connues à l'avance et que des procédures de reprise ne peuvent donc pas être établies par anticipation. L'utilisation de techniques complexes (telles que des systèmes experts, des systèmes à base de règles...) est alors nécessaire pour diagnostiquer et définir des solutions de reprise à partir de la connaissance des états du procédé. Ces procédures ne peuvent donc pas être envisagées comme faisant partie du système de commande.

Nous nous plaçons ici dans le contexte des systèmes pour lesquels il est nécessaire de réagir en temps réel aux évolutions normales et anormales du procédé afin d'assurer la sécurité du matériel et des personnes. Pour cela, il est préférable d'avoir, vu la complexité des systèmes, un ensemble de systèmes de contrôle-commande répartis

qui intègrent localement les fonctions de la surveillance, ceci dans un souci de gain de temps en terme de transfert des informations et de réaction rapide aux anomalies, pour rétablir l'état du procédé.

2.2.2.2 Présentation de l'architecture détaillée

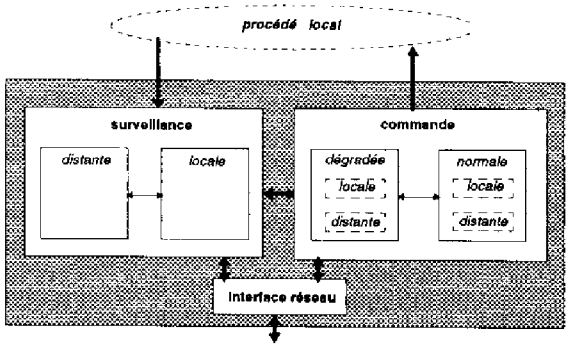

Figure 2: architecture détaillée fonctionnelle d'un système de contrôle-commande local

L'architecture détaillée retenue d'un sous-système de contrôle-commande local est représentée sur la figure 2. Elle comprend :

- un module *surveillance* qui doit réaliser la détection et le diagnostic des anomalies de fonctionnements (dysfonctionnements), ainsi que la décision des reprises, à partir des informations sur les états (normaux et/ou anormaux) du procédé :
  - *local* en mémorisant en temps réel l'ensemble des états du procédé local à partir des comptes-rendus issus des capteurs ;
  - *distant* en mémorisant les états des procédés distants connus grâce aux informations émises par les sous-systèmes de contrôle-commande locaux et reçues via le réseau de terrain.
- un module *commande* qui contient l'ensemble des commandes à exécuter et qui envoie les ordres, soit directement sur le procédé local, soit indirectement sur les procédés distants (à travers le réseau). Deux modes de fonctionnement sont distingués ; ils correspondent à la fonction de commande :
  - *normale* : les commandes normales sont effectives lorsqu'il n'y a pas de défaillances détectées. Les commandes normales peuvent être :
    - *locales* (le plus souvent) et donc destinées au procédé local ;
    - *distantes*, c'est-à-dire envoyées aux procédés distants à travers leur bloc respectif de commande normale.
  - *dégradée* : les commandes relatives aux procédures de reprise à effectuer à la suite de la détection de dysfonctionnements peuvent être soit :

* *locale* afin de contrôler le procédé localement en défaut ;
* *distante* afin de contrôler les procédés distants, en cas de défaillances de leur système de contrôle-commande respectif ;

- un module *interface réseau* qui réalise l'interfaçage entre un sous-système de contrôle-commande et le réseau de communication, permettant ainsi : de recevoir les informations distantes via le réseau (états des procédés distants, ordres émis par des fonctions de commande distante...) ; d'envoyer des informations (ordres pour des fonctions de commande distante, états du procédé local) aux différents sous-systèmes de contrôle-commande.

Nous ne présentons pas ici d'architecture détaillée de l'environnement car seul son comportement externe (interactions) vis-à-vis du système de contrôle-commande est nécessaire.

2.2.3 Les échanges

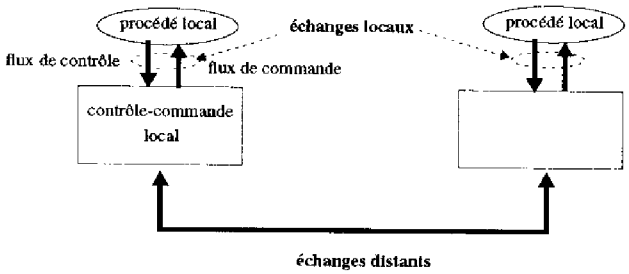

Figure 3: relations entre sites et procédés

Au niveau de l'architecture globale, on doit définir (figure 3) :

- les **échanges locaux** entre un sous-système de contrôle-commande local et son procédé. Ils comprennent des *flux de contrôle* (comptes-rendus issus du procédé) vers le sous-système de contrôle-commande local et des *flux de commande* (commandes envoyées par le contrôle-commande local) vers le procédé.
  La mise en œuvre de ces échanges nécessite de préciser pour chacun : son nom, sa sémantique, l'équipement associé, ses caractéristiques temporelles (liées aux caractéristiques des capteurs et des actionneurs), etc ;

- les **échanges distants** entre les sous-systèmes qui ont des données à échanger via le réseau de communication (entre sites répartis). Ils permettent la réalisation de l'application de contrôle-commande répartie.
  La mise en œuvre de ces échanges nécessite de préciser pour chacun : un nom (identifieur), une sémantique, un type (commande/réponse, alarme, demande de service, signal de synchronisation...), une longueur (important car cela détermine sa durée de transmission), le temps de réponse associé (durée maximale pour la réception d'une réponse), les sites en relation (émetteur/producteur, récepteur/consommateur), la priorité (urgent ou normal), les contraintes temporelles strictes ou relatives (intervalle de

temps maximal toléré pour la mise à disponibilité entre deux sites répartis), la fréquence (périodique/cyclique ou apériodique/acyclique), etc.

Notons que les **échanges internes** (à chaque niveau de décomposition) entre les différents blocs fonctionnels de l'architecture détaillée ne sont pas présentés ici car ils ne sont pas toujours explicités dans les spécifications et ils découlent des décompositions fonctionnelles choisies ; ils doivent permettre de synchroniser les différents modules en relation.

2.3 Réseau de terrain à contrôle d'accès centralisé

2.3.1 L'architecture globale

L'architecture globale d'un réseau de terrain à contrôle centralisé est représentée sur la figure 4 où nous considérons un bus comme médium de transmission. Elle découle directement de l'architecture globale (cf. figure 1) de l'application de contrôle-commande répartie sur un réseau de terrain à contrôle d'accés centralisé.

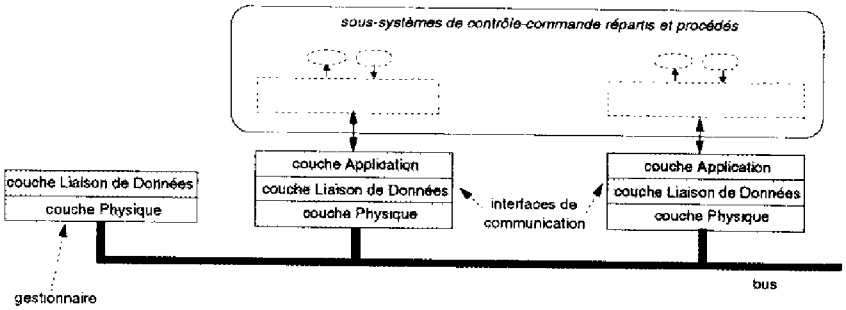

Figure 4: architecture globale d'un réseau de terrain à contrôle centralisé

L'environnement du réseau de terrain comprend les sous-systèmes de contrôle-commande locaux qui y sont connectés, appelés Processus Utilisateurs (PU).

L'architecture globale du réseau de terrain comprend donc :

- les interfaces de communication relatives à chaque sous-système de contrôle-commande local avec les couches application, liaison de données et physique ;
- le gestionnaire d'accès au médium avec les couches liaison de données et physique.

2.3.2 L'architecture détaillée

Nous présentons l'architecture détaillée retenue pour : une Entité Application (EA) et une Entité Liaison (EL) relatives à un site utilisateur (sous-système de contrôle–commande local) ; l'Entité Liaison (EL) relative au site gestionnaire des échanges.

2.3.2.1 Site utilisateur

2.3.2.1.1 Entité Application

La couche application offre aux Processus Utilisateurs (PU) un ensemble de services afin d'accéder aux données locales et/ou distantes (variables, messages, listes, tableaux, évènements de synchronisation...) dans les interfaces de communication. Ces données sont définies par un ensemble d'objets qui possèdent leurs attributs propres (type, status de validité temporelle, en diffusion ou adressé, périodique/apériodique, fréquence...).

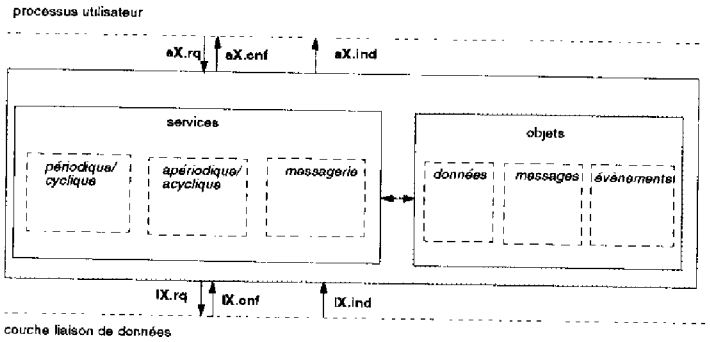

Figure 5: architecture détaillée d'une Entité Application de site utilisateur

L'architecture détaillée retenue pour l'EA d'un site utilisateur est représentée sur la figure 5. Elle comprend :

- un module *services* qui contient l'ensemble des services offerts aux PU (lecture/écriture locales et/ou distantes, demandes de mise à jour, synchronisation...) et l'ensemble des services demandés à l'EL. Ces services peuvent être de type : *périodique/cyclique*, *apériodique/acyclique* pour les échanges de données (variables et/ou messages) lors de la réalisation de l'application de contrôle-commande ou de *messagerie* pour les besoins de la configuration généralement.

  Tous ces services sont obtenus par l'intermédiaire des primitives de services (notées $aX.rq$, $aX.cnf$, $aX.ind$, $lX.rq$, $lX.cnf$, $lX.ind$ où X est le service demandé et les lettres a et l en entête sont relatives respectivement à la couche application ou à la couche liaison) ;

- un module *objets* qui contient l'ensemble des informations relatives aux grandeurs échangées. Ce module peut contenir des objets relatifs à des *données* (variables, listes de variables, tableaux...), des *messages* et/ou des *évènements* (synchronisation, indication...) qui se distinguent des données par le fait qu'ils ne représentent pas une grandeur, mais possèdent une sémantique évènementielle. A chaque objet *donnée* (par exemple, relatif à une température mesurée) sont associés, selon ses attributs spécifiques : un nom, une zone de stockage, le type de stockage (buffer ou file), les droits d'accès, les status temporels, le mode de transmission (périodique ou apériodique), la fréquence (si

périodique)... Notons également qu'à chaque élément *donnée* doivent être associés un nom (du point de vue de l'utilisateur) et un identifieur (du point de vue du réseau), définis uniques parmi toutes les données de l'application.

2.3.2.1.2 Entité Liaison

D'après le standard IEEE 802.2 (Logical Link Layer) sur les réseaux locaux, deux sous-couches liaison doivent être définies : la sous-couche LLC (Logical Link Control) et la sous-couche MAC (Medium Access Control). Cependant, soulignons que ces sous-couches ne sont pas aussi explicitement définies dans toutes les normes de réseaux locaux et de réseaux de terrain, mais leurs fonctionnalités doivent toujours être offertes.

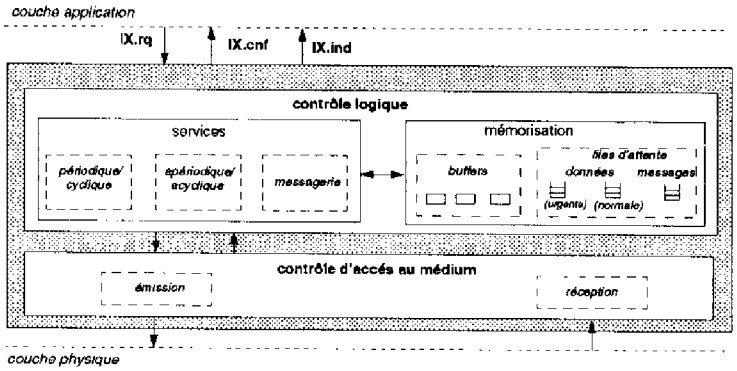

Figure 6: architecture détaillée d'une Entité Liaison de site utilisateur

L'architecture détaillée retenue pour une Entité Liaison (EL) d'un site utilisateur est représentée sur la figure 6. Elle comprend :

- un module *contrôle logique* qui assure la réalisation des services à fournir à la couche application. Le module peut être décomposé en :
  - un module *services* qui assure la gestion des services offerts à la couche application et qui sont obtenus au moyen des primitives $lX.rq$, $lX.cnf$, $lX.ind$;
  - un module *mémorisation* qui mémorise à la fois les données reçues de la couche application et celles reçues via le réseau. Les données sont stockées dans : des *buffers* ou des *files d'attente* relatives à des données *urgentes ou normales*, ou des *messages* ;
- un module *contrôle d'accès au médium* qui assure la gestion de l'accès au bus en *réception* et en *émission* des trames.

2.3.2.2 Site gestionnaire centralisé d'accès au bus

Dans le contexte des réseaux locaux à accès centralisé, le site gestionnaire a la gestion de tous les droits d'accès au bus. Il a donc deux fonctions essentielles : établir l'ordonnancement des échanges et assurer l'accès au médium.

L'architecture détaillée retenue pour une Entité Liaison de site gestionnaire est représentée sur la figure 7. Elle comprend :

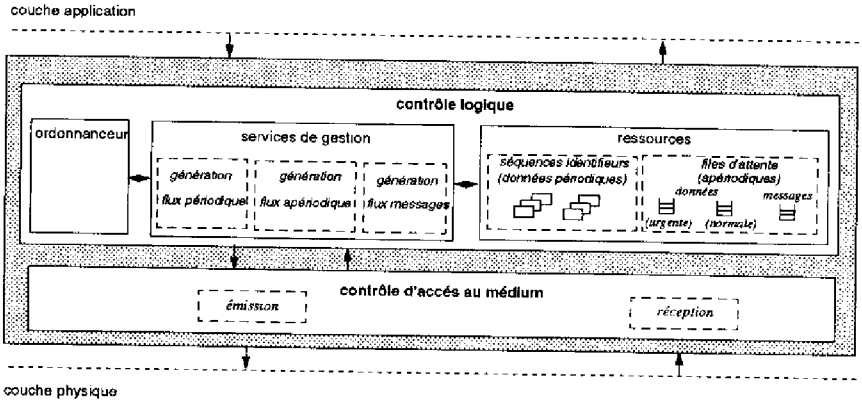

Figure 7: architecture détaillée de l'Entité Liaison du site gestionnaire

- un module *contrôle logique* qui comprend :
  - un module *services de gestion* d'accès (arbitrage) permettant de gérer et générer les séquences de trames relatives aux :
    - *flux périodique.* Ce flux implique généralement que la plupart des échanges soient pré-définis (déterminés) et ordonnés dans une table d'ordonnancement (ou de scrutation) en fonction des besoins de l'application (ceci correspond à la fonction de Distributeur dans le cadre des communications de type Producteur/Consommateur/Distributeur [TN89]) ;
    - *flux apériodique.* Ce flux est lié aux requêtes d'échanges des processus utilisateurs et il est généralement inséré dans les temps morts du trafic périodique ou dans les temps réservés aux échanges apériodiques (ceci correspond, en quelque sorte, à la gestion des échanges de type Client/Serveur) ;
    - *flux de messages.* Ce flux est lié aux requêtes de messages envoyées par les PU ;
  - un module *ressources* qui regroupe l'ensemble des ressources nécessaires à la réalisation des échanges. Il comprend :
    - des *files d'attente* contenant les éléments pour les requêtes d'échanges de données de priorité *normale* ou *urgente*, ainsi que pour les requêtes de transfert de

messages. Ces échanges seront satisfaits au cours du trafic périodique et/ou apériodique (selon le temps disponible) ;

* des *séquences d'identifieurs* de données à diffuser périodiquement pour leur mise à jour ;

– un module *ordonnanceur* qui contient un (des) algorithme(s) d'ordonnancement, afin de déterminer l'ordonnancement des échanges d'un ensemble de données (variables et messages) qui ont des contraintes temporelles strictes ou relatives à satisfaire. L'ordonnancement doit pouvoir être effectué hors ligne (dans le cas où le trafic est bien connu) et/ou en ligne (utile en cas de modifications des échanges) si de nouvelles contraintes temporelles sont imposées aux échanges de données/messages ;

- un module *contrôle d'accès au médium* qui gère les accès au médium :
  – en *émission* des trames pour donner la parole aux PU. Son rôle est de délivrer un indicateur de droit de parole alternativement aux différentes stations connectées sur le médium ;
  – en *réception* des trames relatives à des données échangées et/ou à des requêtes issues des stations réparties.

2.3.3 Les échanges

Il est nécessaire de préciser la nature des échanges à effectuer entre les différentes interfaces de communication pour la réalisation des échanges distants répondant aux besoins de communication de l'application répartie. Il sera donc nécessaire de déterminer :

- les **échanges locaux**, i.e. les **services**, dans chaque interface entre les différentes couches (Processus Utilisateur/couche Application et couche Application/couche Liaison) pour la réalisation des échanges de données et de messages périodiques, apériodiques, à contraintes temporelles strictes ou relatives... Les contraintes temporelles de ce type d'échanges dépendent des techniques d'implantation utilisées (compilateur, algorithme d'ordonnancement...) ;

- les **échanges distants**, i.e. les **protocoles**, entre les interfaces de communication des sites utilisateurs et du site gestionnaire. Ces échanges doivent respecter les contraintes temporelles qui découlent d'une part, des différents types de trafic que nécessite le déroulement correct de l'application (trafic périodique, trafic apériodique,...) et d'autre part, des durées des temps de transfert de ces trafics (cf. paragraphe 2.2.3).

Notons que les échanges résultant du raffinement interne des entités de communication entre les différents modules ne sont pas présentés ici ; ils dépendent de la manière d'implanter les synchronisations internes et des techniques utilisées pour la modélisation.

3 Modélisation

La taille et la complexité des systèmes à étudier nous ont amené à considérer leur structuration pour une décomposition fonctionnelle et à préciser leur architecture globale et détaillée en modules élémentaires, ainsi que les échanges. Cependant, la problématique réside maintenant dans la description des comportements dynamiques de chaque module élémentaire au moyen des Réseaux de Petri Temporisés Stochastiques. Pour répondre à cette problématique, cette partie présente donc :

- des principes pour la modélisation, au moyen des Réseaux de Petri Temporisés Stochastiques, de comportements dynamiques à caractéristiques temporelles ;
- des structures de modèles en RdPTS d'expressions de comportements simples (*et, ou...*) et complexes (*il existe, tout est faux, tout n'est pas vrai...*) ayant des caractéristiques temporelles et issues directement des spécifications initiales exprimées en langage naturel [AFN90b] [Sab93a]. Ces structures vont permettre de faciliter la modélisation.

3.1 Que faut il modéliser ?

Nous proposons de construire un ensemble de petits modèles élémentaires, relatifs au comportement de chaque module élémentaire, puis de les composer en vue de la construction du modèle complet du système. Cependant, ceci nécessite de définir des principes de modélisation et de composition en RdPTS à respecter pour garantir que les évolutions temporelles des éléments du système global ne soient pas remises en cause par la composition.

A partir des spécifications initiales d'un système, il est nécessaire d'isoler pour la modélisation de chaque module élémentaire :

- les éléments tels que :
  - les états qui représentent les activités du module ;
  - les conditions internes et externes. Les conditions internes représentent les valeurs et/ou attributs des objets (données de type variable, message...) manipulés par le module considéré. Les conditions externes portent sur des états et/ou des conditions d'autres modules élémentaires du même sous-système ;
  - les échanges de messages entre ce module et les modules élémentaires des sous-systèmes adjacents. Ces messages sont généralement spécifiés explicitement dans le cahier des charges. Ils véhiculent une information ;
- le comportement dynamique (règles d'évolutions) d'un module élémentaire qui s'exprime en termes de changements d'états et/ou de conditions dépendant des évolutions des conditions internes et/ou externes, ainsi que des interactions.

3.2 Principes fondamentaux de modélisation et de composition

3.2.1 Modélisation d'un module élémentaire

3.2.1.1 Nos principes de modélisation

Les modèles basés sur les réseaux de Petri sont bien adaptés pour la représentation des comportements dynamiques, mais ils sont relativement mal adaptés à la représentation des données (sauf cas particulier). Cependant, comme nous nous situons dans le contexte d'étude de systèmes de contrôle-commande et de réseaux de terrain où la plupart des informations peuvent être considérées de type booléen, ceci nous amène à poser le principe 1.

Principe 1 *: Nous caractériserons les données, et leurs attributs de type booléen (vrai/faux, 0/1,...) nécessaires à la modélisation du comportement dynamique d'un module comme une condition (proposition) logique qui possède toujours sa condition (proposition) complémentaire.*

Une temporisation peut également être considérée comme une condition logique booléenne dont les deux valeurs sont relatives à son état actif ou non, ce qui nous amène (de par le principe 1) à poser le principe 2.

Principe 2 *: Nous caractériserons une temporisation comme une condition logique qui possède sa condition complémentaire. La temporisation est : soit en cours (notée On), soit au repos (notée Off).*

De plus, il est important, afin d'exploiter la modularité, d'adopter un principe de modélisation qui permette de vérifier le comportement de chaque module élémentaire. Ceci est essentiel dans le contexte des systèmes complexes temps réel répartis, afin de localiser et de corriger au plus près des erreurs de conception, ce qui nous amène à poser le principe 3.

Principe 3 *: Le comportement dynamique d'un module élémentaire, en ne considérant pas ses liens avec les autres modules avec lesquels il est en relation, doit être cyclique et doit avoir de bonnes propriétés (borné, autonome..). De plus, il doit être construit de façon à prévoir l'ensemble de ses évolutions en fonction de ses intéractions avec son environnement.*

3.2.1.2 Etablissement du modèle

A partir des éléments et des évolutions identifiés précédemment, on associe à chaque module élémentaire M_i :

- l'ensemble E_i de ses états ;
- l'ensemble C_i des conditions booléennes relatives à ses données et ses temporisations, ainsi que l'ensemble $\overline{C_i}$ des conditions complémentaires des éléments de C_i ;
- l'ensemble EV_i des interactions (évènements) entre le module M_i et les autres modules (du même sous-système et/ou de sous-systèmes adjacents) ;

- les ensembles EE_i, CE_i et $\overline{CE_i}$ qui représentent respectivement les états, conditions et conditions complémentaires Externes, c'est-à-dire des autres modules élémentaires M_j $(\forall j \neq i)$ qui conditionnent les évolutions du module élémentaire M_i considéré.

Le modèle, en Réseaux de Petri Temporisés Stochastiques, qui représente le comportement dynamique d'un module élémentaire sera donc construit tel que :

- une place est associée à chaque élément des ensembles E_i, C_i, $\overline{C_i}$, EE_i, CE_i, $\overline{CE_i}$. Nous appelons les places intrinsèques, les places des ensembles E_i, C_i, $\overline{C_i}$ du module élémentaire et nous les notons p. Nous appelons les places extrinsèques, les places des ensembles EE_i, CE_i, $\overline{CE_i}$, importées des autres modules et nous les notons $M_j[p]$ où M_j est le nom de leur module d'appartenance ;

- une transition t est associée :

  - à chaque changement d'état provoqué par un évènement de EV_i. Ces interactions sont représentées en utilisant la notation introduite par Hoare [Hoa78] où $c!ev_i$ et $c?ev_i$ représentent respectivement l'émission et la réception sur le canal de communication c du message ev_i (dans notre contexte où il n'y a pas d'ambiguïtés sur le canal, il ne sera pas précisé) ;
  - à chaque changement d'états et/ou de conditions lié à des évolutions internes au module et/ou à d'autres modules (expiration d'une temporisation, changement de valeur d'une condition, fin d'une action temporisée,...) ;

- un intervalle de temps initial i_0 est associé à chaque transition tel que $i_0 = [a, a]$ où a est un réel positif ou nul. Cette hypothèse restrictive de définition des intervalles temporels permet de représenter les éléments essentiels dans cette étude que sont les actions de durée non nulle et les temporisations de durée constante ;

- une fonction de densité de probabilité initiale f_0 est associée à l'intervalle de temps initial de chaque transition. L'hypothèse précédente sur l'intervalle i_0 implique, par conséquent, que les densités de probabilité sont toutes déterministes telles que $f_o(x) = \delta(x-a)$ (impulsion de Dirac) où : si $a \neq 0$, cela représente une action de durée non nulle et/ou la durée d'une temporisation ; si $a = 0$, cela représente une transition immédiate de durée nulle ;

- un marquage initial est associé à l'ensemble des places qui représentent : l'état initial du module (qui est un état d'accueil) et les états initiaux des conditions internes booléennes (temporisées ou non).

Remarque : par la suite, nous indiquerons uniquement les distributions temporelles associées aux transitions non immédiates ; par défaut les autres transitions seront considérées immédiates.

3.2.2 La problématique du test de conditions et/ou d'états temporisés

Le test d'une condition (variable booléenne ou temporisation) vraie ou fausse, respectivement d'un état actif, en Réseaux de Petri Temporisés Stochastiques peut être effectué par des accès en lecture sur des transitions immédiates de deux manières différentes :

- soit par le test à un de la place représentant la condition temporisée vraie (figure 8.a), respectivement l'état actif (figure 9.a) ;
- soit par le test à zéro, au moyen d'arcs inhibiteurs, de la place représentant la condition complémentaire fausse (figure 8.b), respectivement des places représentant les états complémentaires non actifs (figure 9.b).

Rappelons que les arcs inhibiteurs ont été introduits [Bra83] dans les réseaux de Petri à la suite de la constatation que l'on ne pouvait pas exprimer directement qu'une évolution est soumise à la non vérification d'une condition. Or, l'introduction des places représentant la condition complémentaire (dans le cas des réseaux bornés) permet de tester facilement cette condition. Notons toutefois que des analyses structurelles ne seront pas permises, mais celles-ci ne font pas partie de nos objectifs d'analyse.

L'inconvénient majeur d'effectuer des accès en lecture par le test à un sur des places marquées en Réseaux de Petri Temporisés Stochastiques où toutes les transitions sont temporisées **est, que si le test est effectué, alors il ya désensibilisation des éventuelles transitions temporisées concurrentes de durée non nulle.**

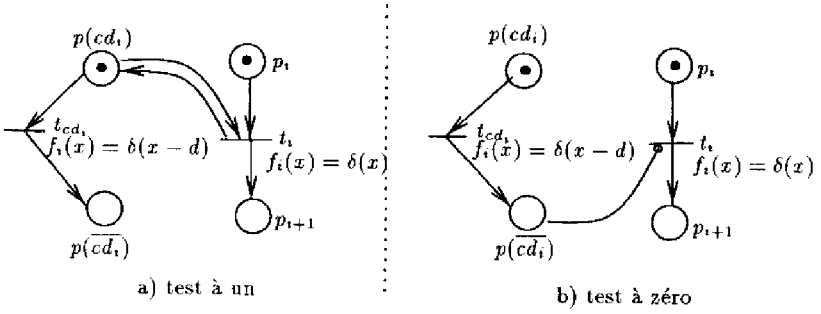

Figure 8: test de condition temporisée

La figure 8.a montre le test à un de la place $p(cd_i)$ qui représente l'état actif d'une condition temporisée à partir de laquelle la transition t_{cd_i} de durée d non nulle ($f(x) = \delta(x-d)$) est sensibilisée. Le test à un de cette place (tir de la transition t_i immédiate) va désensibiliser la transition t_{cd_i}. La figure 8.b montre le test à zéro de la place $p(\overline{cd_i})$ représentant l'état non actif de la temporisation ; dans ce cas, la transition temporisée t_{cd_i} n'est pas désensibilisée.

De la même façon, la figure 9.a montre le test à un de l'état actif p_{i+2} d'une structure de contrôle représentée par les trois états p_i, p_{i+1} et p_{i+2}. Le test à un de cette dernière place (tir de la transition t_j) va désensibiliser la transition t_{i+2}. La figure 9.b montre le test à zéro de l'ensemble des états complémentaires, évitant ainsi la désensibilisation de t_{i+2}.

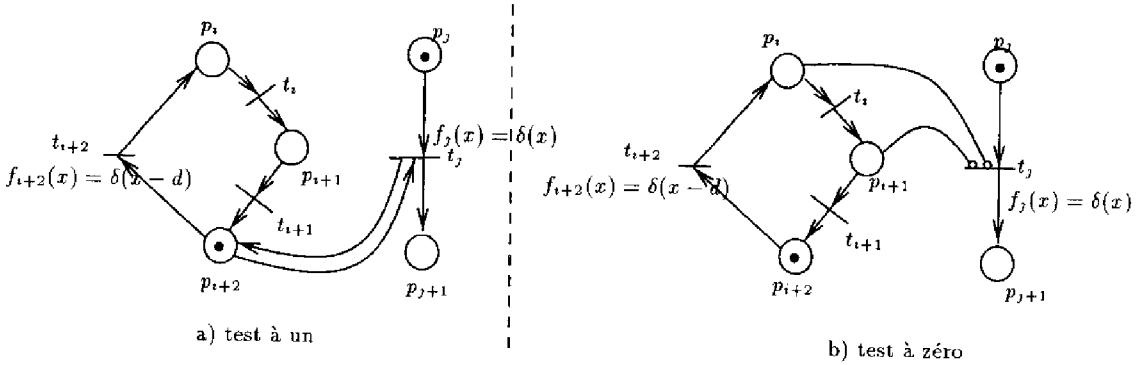

Figure 9: tests d'états

Ces considérations montrent tout l'intérêt de l'introduction des conditions complémentaires qui permettent d'effectuer le test à zéro sur les places complémentaires en utilisant les arcs inhibiteurs.

3.2.3 Construction des modules élémentaires

Compte tenu de la modélisation modulaire (dont l'objectif est de faciliter la modélisation), la composition des modules élémentaires (dans un sous-système et/ou entre sous-systèmes adjacents) est une phase importante pour la réalisation des interactions entre les différents modules. La synchronisation entre modules élémentaires est représentée en utilisant la notation ! et ? pour signifier respectivement l'émission et la réception d'un message.

3.2.3.1 Sur les modes de construction des réseaux de Petri

Les différents modes de composition des modèles de réseaux de Petri qui existent sont : la fusion de transitions ; la fusion de places [FP94] et les places de communication (ou partagées).

Fusion de transitions

La composition par fusion de transitions correspond en réseaux de Petri à la composition de modèles sur l'ensemble des transitions qu'ils ont en commun. Il s'agit d'une synchronisation forte entre les éléments communicants (i.e. sémantique du rendez-vous de Milner [Mil80] entre deux processus communiquants) : les deux modules doivent être prêts simultanément pour effectuer le rendez-vous qui est atomique.

Cependant, la fusion de deux transitions temporisées en rendez-vous a pour inconvénient de nécessiter la présence des jetons dans les places d'entrée des transitions en rendez-vous pendant toute la durée spécifiée par les distributions temporelles pour permettre leur tir (mais ce mécanisme est non bloquant). De plus, le tir de transitions concourantes peut les désensibiliser. Les avantages sont cependant de minimiser les modèles en termes de nombres de places et de transitions et ainsi, faciliter l'analyse en terme de nombre d'états du graphe

d'états probabilisé.

La fusion de deux transitions temporisées, t_i et t_j, est permise en Réseaux de Petri Temporisés Stochastiques lorsque les transitions en rendez-vous ont des distributions identiques et quelles sont sensibilisées simultanément :
– soit immédiates telles que $f_i(x) = f_j(x) = \delta(x)$ d'où $f_{i,j}(x) = \delta(x)$ après composition ;
– soit temporisées de durée d identique telles que $f_i(x) = f_j(x) = \delta(x - d)$ d'où $f_{i,j}(x) = \delta(x - d)$ après composition.

Fusion de places

La composition par fusion de places dupliquées correspond, en réseaux de Petri, à la composition de modèles sur l'ensemble des places qu'ils ont en commun. Les places, qui appartiennent à un module (unique) peuvent être utilisées par d'autres modules en particulier, pour des accès en lecture. Comme précisé dans la problématique du test (cf. § 3.2.2), la fusion de places utilisées pour le test à un a l'inconvénient majeur de risquer de désensibiliser des transitions temporisées ; cet inconvénient, comme on l'a vu, disparaît par l'utilisation du **test à zéro.**

Il n'y a alors aucune contrainte sur les spécifications temporelles associées aux transitions.

Places de communication

La composition par places de communication consiste à introduire des places entre les transitions en rendez-vous. La synchronisation par place de communication unique (boîte aux lettres) correspond à une file d'attente. D'autres schémas de composition asynchrone par places peuvent être définis, tels que la procédure d'appel/réponse ; ceci nécessite de définir de nouveaux messages relatifs à l'appel (*!appel/?appel*) et à la réponse (*!reponse/?reponse*), donc aussi des places relatives, respectivement à l'envoi de l'appel par l'émetteur et à l'envoi de la réponse par le récepteur, ainsi que des transitions intermédiaires. Ce mode de communication est qu'il peut être bloquant : pour le récepteur en attente du message *?appel* si l'émetteur ne l'a pas encore envoyé ; pour l'émetteur en attente du message *?reponse* si le récepteur ne l'a pas encore envoyé. De plus, ce mode de composition augmente la taille des modèles et donc complexifie (pénalise) l'analyse (en augmentant la taille des graphes d'accessibilité). Cependant, l'avantage est de réaliser des communications asynchrones entre les modèles.

Quelque soit le schéma de composition par places de communication, il n'y a aucune contrainte sur les spécifications temporelles associées aux transitions en rendez-vous (de par les places qui introduisent un degré de liberté).

3.2.3.2 Modes de constuction de nos modèles

Suite à l'ensemble de ces considérations et **compte tenu du contexte temps réel de l'étude et des spécificités des RdPTS**, nous considérons qu'**il est nécessaire de se prémunir contre les désensibilisations et les réinitialisations intempestives de transitions temporisées lors de la construction modulaire.** Nous introduisons donc les principes 4 et 5 pour la construction des modules élémentaires modélisés au moyen des Réseaux de Petri Temporisés Stochastiques.

Principe 4 *: La construction de modules élémentaires d'un même sous-système est effectuée uniquement par fusion des transitions (immédiates et simultanément sensibilisées) et/ou par fusion de places, en utilisant spécifiquement le test à zéro lors des accès en lecture.*

Ce mode de composition entre les modules élémentaires d'un même sous-système permet de les faire évoluer de façon synchrone et atomique ; ce schéma de construction sera largement utilisé par la suite. Cela revient à considérer que les échanges internes (à l'intérieur d'un même sous-système) sont uniquement effectués par synchronisation forte et par variables partagées.

Principe 5 *: Les interactions entre modules élémentaires de sous-systèmes distincts adjacents seront effectuées par places de communication.*

Ce mode de composition permet de préserver l'asynchronisme des évolutions comportementales entre modules de sous-systèmes adjacents et de représenter explicitement des transferts de messages.

3.2.4 Structures élémentaires temporisées

Dans le contexte de la modélisation des systèmes temps réel, la prise en compte du temps est primordiale. Nous distinguons ici deux types de spécifications temporelles : les **temporisations** et les **actions de durée non nulle.** Leur différence est fondamentale : une temporisation correspond à un délai, tandis qu'une action temporisée représente une action du système qui dure un certain temps.

3.2.4.1 Représentation et test d'une temporisation

Une temporisation est considérée comme une condition booléenne qui peut prendre les deux états complémentaires : en cours (*on*) ou au repos (*off*). La représentation d'une temporisation devra comprendre au miminum les deux places relatives à ces deux états.

Dans notre étude, cette représentation sera largement utilisée ; elle constitue donc une des représentations élémentaires nécessaire à la modélisation de systèmes temps réel en RdPTS.

La figure 10 représente le test d'une temporisation en cours par son test à zéro, afin de ne pas la désensibiliser. Le module $M(tmp)$ modélise la condition *temporisation* de durée d qui est en cours car $p(on)$ est marquée suite au tir de t_{on}. La transition t_{off} temporisée de durée

d est sensibilisée. Le module M_i modélise une activité en cours par le marquage de la place p_i qui, par le tir de t_i, atteint la place p_{i+1} de l'état suivant si la temporisation est en cours, c'est-à-dire si elle n'est pas au repos (pas de jeton dans $p(off)$).

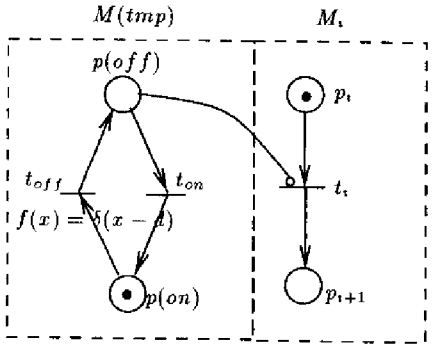

Figure 10: temporisation

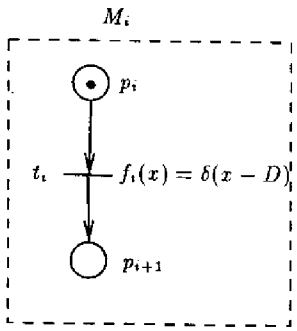

Figure 11: action temporisée

3.2.4.2 Représentation d'une action temporisée

Une action temporisée est considérée comme une activité d'un module (partie d'un module) qui dure un certain temps fixé et non nul. Dans les spécifications, cela peut correspondre à un temps de manœuvre ou un temps de réponse d'un élément du système.

La figure 11 représente une partie du module M_i : celui-ci est dans l'état p_i et exécute une action temporisée de durée D avant de passer dans l'état suivant p_{i+1} après le tir de t_i, dont la fonction de densité associée est $f_i(x) = \delta(x - D)$.

3.3 Des spécifications informelles à la formalisation en Réseaux de Petri Temporisés Stochastiques

3.3.1 Motivations

Généralement, les descriptions des comportements dynamiques dans les cahiers des charges, sont exprimées par l'énoncé des spécifications en *langage naturel*. La formalisation de ces spécifications est donc essentielle dès les premières phases de développement afin de préciser de manière non ambiguë les comportements et ainsi, permettre de valider le comportement du système (vérifier que ce qui est modélisé répond aux propriétés attendues) en utilisant des techniques d'analyse formelles.

Or, décrire formellement un système à partir des spécifications exprimées en langage naturel dans le cahier des charges constitue une étape délicate et importante pour la réussite des étapes ultérieures.

C'est pourquoi, pour répondre à la question "*comment décrire formellement en RdPTS le comportement d'un système à partir des spécifications initiales en langage naturel ?* ", nous

proposons des schémas de représentations d'assertions logiques élémentaires (telles que *et*, *ou*, *ou exclusif*) sous forme de Réseaux de Petri Temporisés Stochastiques en utilisant les principes de modélisation (1 à 3) et de composition (4 et 5) présentés précédemment dans la partie 3.2, ainsi que des schémas de représentations d'assertions complexes (telles que *tout... est vrai*, *tout... est faux*, *il existe...*, *il n'existe pas...*) à partir des schémas d'assertions élémentaires.

3.3.2 Assertions logiques élémentaires

3.3.2.1 Notations

Nous utilisons les opérateurs de la logique $\wedge$, $\vee$ et $\oplus$ pour exprimer respectivement les relations ET, OU et $OU\_EXclusif$. La modélisation en réseaux de Petri de la relation $\wedge$ est représentée par la réunion sur la même transition de l'ensemble des arcs entrants de toutes les pré-conditions à vérifier simultanément. La relation OU est représentée par l'union des transitions compétitives dont les arcs entrants sont respectivement issus de chacune des pré-conditions vérifiées. La relation $\oplus$ est représentée par l'union des transitions en compétition dont les arcs entrants sont issus des conditions vérifiées et des autres conditions non vérifiées.

Or, comme déjà précisé dans un contexte temps réel, afin de ne pas désensibiliser des transitions temporisées, les accès en lecture sont effectuées en RdPTS par le test à zéro au moyen d'arcs inhibiteurs **sur les propositions complémentaires**. Pour cela, chaque proposition affirmative doit être transformée en sa proposition complémentaire niée au moyen de l'opérateur de négation $\neg$.

3.3.2.2 ... ET...

Lorsque les spécifications précisent que le changement d'état, de l'état courant p_i à l'état suivant p_{i+1}, d'un module M_i est basé sur la présence de deux conditions cd_i ET cd_j simultanément vraies (pouvant appartenir chacune à des modules distinct $M(cd_i)$ et $M(cd_j)$), la pré-condition de tir de t_i de M_i est notée $cd_i \wedge cd_j$.

"Soient cd_i, $cd_j \in C$, cd_i ET cd_j sont vraies simultanément..."

$$\equiv$$

$$cd_i \wedge cd_j \tag{2.1}$$

$$\Longleftrightarrow \neg(\overline{cd_i}) \wedge \neg(\overline{cd_j}) \tag{2.2}$$

La modélisation en RdP de l'équation 2.1 (resp. 2.2) est représentée sur la figure 12.a par le test à un des conditions simultanément vraies pour le tir de t_i (resp. figure 12.b par le test à zéro des conditions complémentaires simultanément fausses).

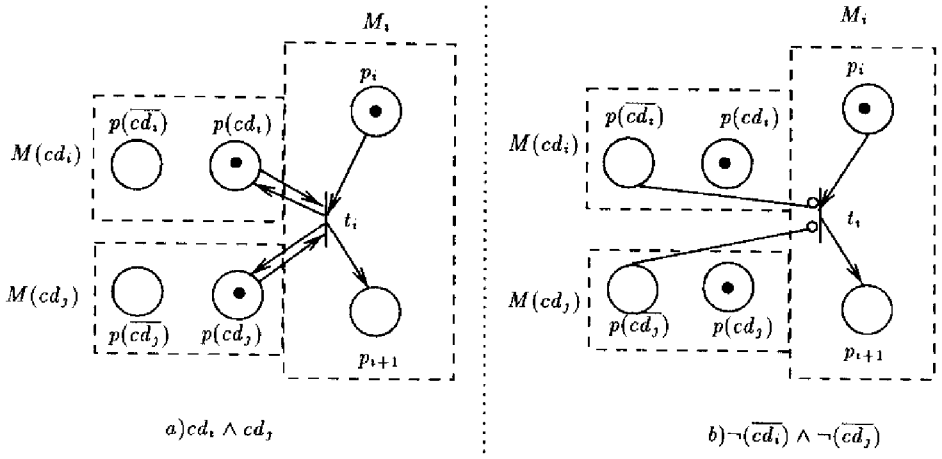

Figure 12: cd_i et cd_j sont vraies

3.3.2.3 ... OU ...

Lorsque les spécifications précisent que le changement d'état, de l'état courant p_i à l'état suivant p_{i+1}, du module M_i, est provoqué par l'une des deux conditions cd_i *OU* cd_j vraie, la condition d'évolution est notée $cd_i \vee cd_j$.

"*Soient* cd_i, $cd_j \in C$, cd_i *OU* cd_j *est vraie...*"

$$\equiv$$

$$cd_i \vee cd_j \tag{2.3}$$

$$\Longleftrightarrow \neg(\overline{cd_i}) \vee \neg(\overline{cd_j}) \tag{2.4}$$

La modélisation en RdP de l'équation 2.3 (resp. 2.4) est représentée figure 13.a (resp. 13.b) par les deux transitions t_i et t'_i.

3.3.2.4 ... OU exclusif...

Lorsque les spécifications précisent que le changement d'état, de l'état courant p_i à l'état suivant p_{i+1}, du module M_i, est provoqué par le choix exclusif entre deux (ou plusieurs) conditions, ceci signifie que le module change d'état si et seulement si, simultanément une des conditions est vraie et pas l'autre.

"*Soient* cd_i, $cd_j \in C$, cd_i `OU EXclusivement` cd_j *est vraie...*"

$$\equiv$$

$$cd_i \oplus cd_j \Longleftrightarrow (cd_i \wedge \overline{cd_j}) \ \vee \ (\overline{cd_i} \wedge cd_j) \tag{2.5}$$

$$\Longleftrightarrow (\neg\overline{cd_i} \wedge \neg cd_j) \vee (\neg cd_i \wedge \neg\overline{cd_j}) \tag{2.6}$$

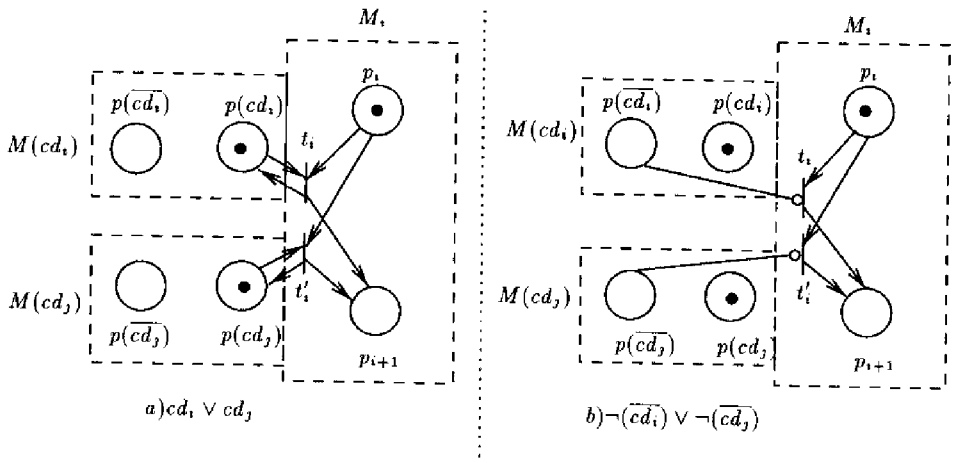

Figure 13: cd_i OU cd_j est vraie...

La modélisation en RdP de l'équation 2.5 (resp. 2.6) est représentée sur la figure 14.a (resp. figure 14.b) par les transitions t_i et t'_i.

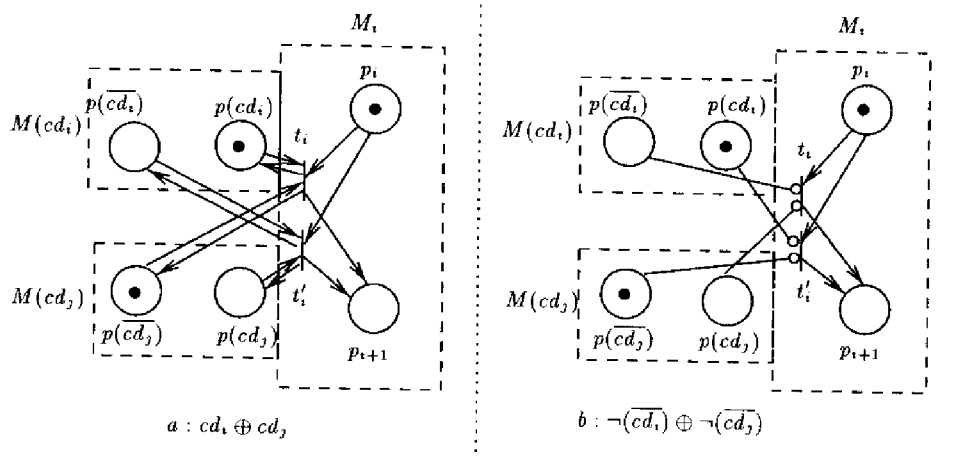

Figure 14: cd_i ou exclusivement cd_j est vraie...

3.3.3 Assertions affirmatives

3.3.3.1 Tout... est vrai

Lorsque les spécifications précisent qu'un module évolue, de l'état courant p_i à l'état suivant p_{i+1}, lorsque toutes les conditions cd_v sont vraies, alors elles doivent toutes être vraies simultanément.

Considérons V, le sous-ensemble de C contenant les conditions qui doivent être vraies, alors $V = \{ cd_v \mid cd_v = 1 \; et \; cd_v \in C \}$.

Soient $cd_1, cd_2, ..., cd_v \in V$ *"toutes les* cd_v *sont vraies..."*

$$\equiv$$

$$cd_1 \wedge cd_2 \wedge ... \wedge cd_v \Longleftrightarrow \prod_{V} cd_v \tag{2.7}$$

$$\Leftrightarrow \neg\overline{cd_1} \wedge \neg\overline{cd_2} \wedge ... \wedge \neg\overline{cd_v} \Longleftrightarrow \prod_{\overline{V}} \neg\overline{cd_v} \tag{2.8}$$

La modélisation en RdP de l'équation 2.7 (resp. 2.8) est représentée sur la figure 15.a (resp. 15.b) par la transition t_i.

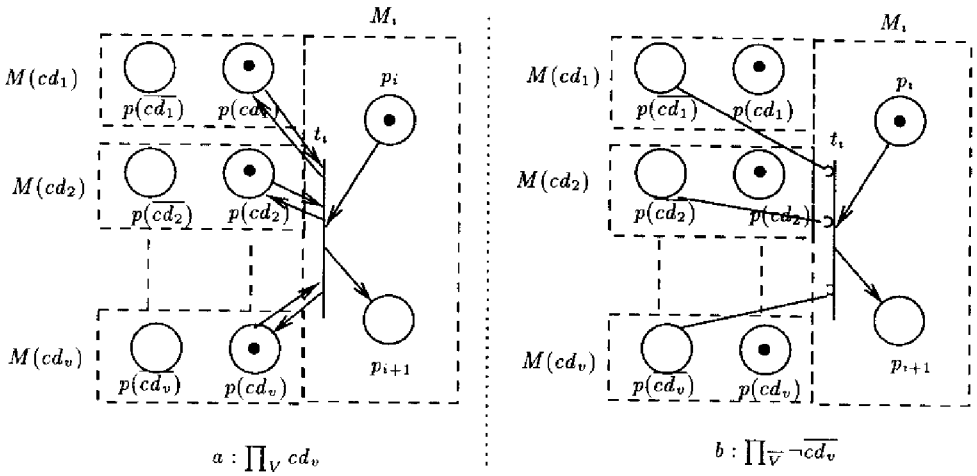

Figure 15: toutes les cd_v sont vraies...

3.3.3.2 Tout... est faux

Lorsque les spécifications précisent que le module M_i évolue, de l'état courant p_i à l'état suivant p_{i+1}, si toutes les conditions cd_f sont fausses, alors elles doivent toutes être fausses simultanément. Soit F, le sous-ensemble des conditions cd_f à vérifier : $F = \{\, cd_f | cd_f = 0 \; et \; cd_f \in C \}$

Soient $cd_1, cd_2, ..., cd_f \in F$, *"toutes les* cd_f *sont fausses..."*

$$\equiv$$

$$\prod_{\overline{F}} \overline{cd_f} \tag{2.9}$$

$$\Longleftrightarrow \prod_{F} \neg cd_f \tag{2.10}$$

La modélisation de l'équation 2.9 (resp. 2.10) est montrée sur la figure 16.a (resp. figure 16.b) par la transition t_i.

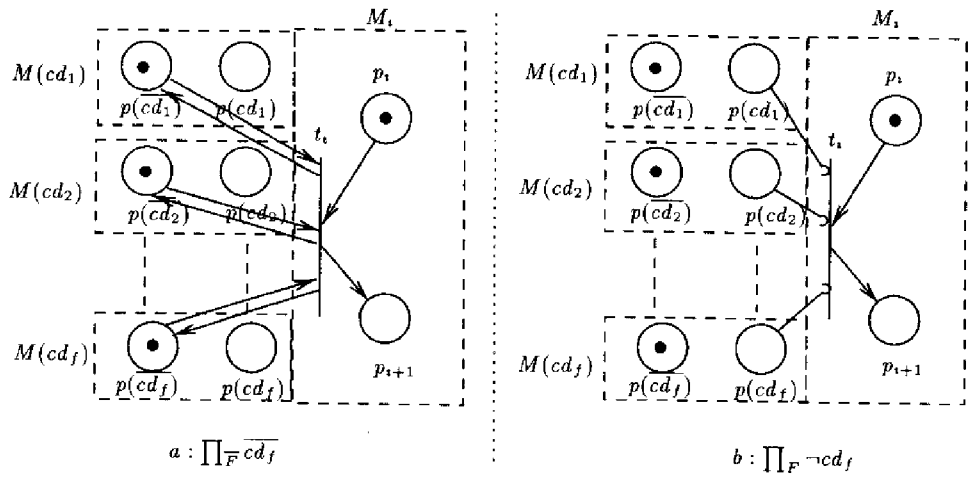

Figure 16: toutes les cd_f sont fausses...

3.3.3.3 Il existe au moins...

Lorsque le comportement du module M_i évolue, de l'état courant p_i à l'état suivant p_{i+1}, s'il existe une condition cd_i vraie parmi les conditions cd_x d'un sous-ensemble X de conditions, alors il doit exister au moins une condition cd_i vraie, $X = \{ cd_x | \exists cd_i = 1\}$

Soient $cd_1, cd_2, ..., cd_i, ..., cd_x \in X$, " *Il existe au moins une* cd_i *vraie parmi les* cd_x"

$$\equiv$$

$$cd_1 \vee cd_2 ... \vee cd_i \vee ... \vee cd_x \Longleftrightarrow \sum_X cd_x \tag{2.11}$$

$$\Longleftrightarrow \sum_{\overline{X}} \neg \overline{cd_x} \tag{2.12}$$

La modélisation en RdP de l'équation 2.11 (resp. 2.12) est représentée sur la figure 17.a (resp. figure 17.b) par l'ensemble des transitions t_1, t_2,..., t_x ($x = card(X)$).

Remarque : par extrapolation, la relation *il existe au plus...* est représentée sur la figure 18.a par le test à un et sur la figure 18.b par le test à zéro.

3.3.4 Assertions négatives

3.3.4.1 Tout... n'est pas faux

Le module M_i évolue, de l'état courant p_i à l'état suivant p_{i+1}, si parmi les conditions cd_x de X, toutes ne sont pas fausses : c'est-à-dire s'il en existe au moins une de vraie dans X.

Soient $cd_1, cd_2, ..., cd_i, ..., cd_x \in X$, " *Toutes les* cd_x *de* X *ne sont pas fausses*"

$$\equiv$$

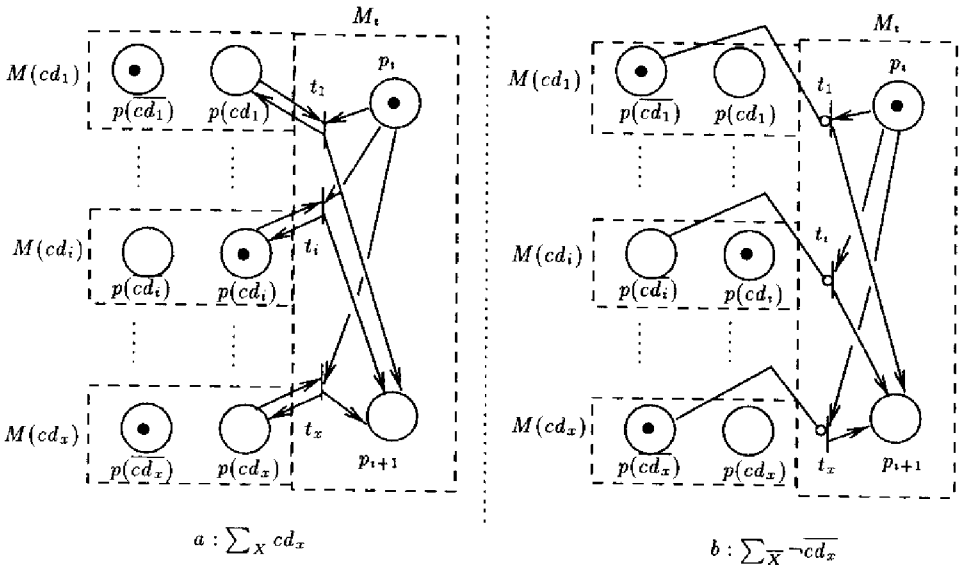

Figure 17: il existe au moins une cd_i vraie...

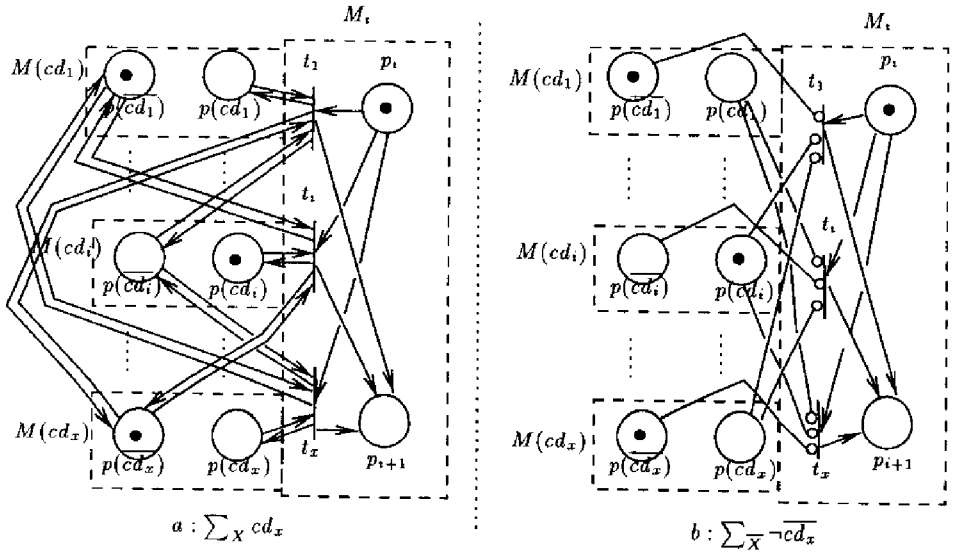

Figure 18: il existe au plus une cd_i vraie...

" *Il existe au moins une* cd_i *vraie parmi les* cd_x *de* X"

$$\equiv$$

$$\neg(\prod_{\overline{X}} \overline{cd_x}) \Longleftrightarrow \sum_{X} cd_x \tag{2.13}$$

$$\Longleftrightarrow \sum_{\overline{X}} \neg\overline{cd_x} \tag{2.14}$$

La représentation est identique à celle de la figure 17.

3.3.4.2 Tout... n'est pas vrai

Le module M_i évolue de l'état *pré* à l'état *post* si toutes les conditions cd_x de X ne sont pas vraies, c'est-à-dire s'il existe au moins une condition cd_x fausse dans X.

" *Toutes les* cd_x *de* X *ne sont pas vraies*"

$$\equiv$$

" *Il existe une* cd_i *fausse parmi les* cd_x *de* X "

$$\equiv$$

" *Il existe une* $\overline{cd_i}$ *vraie parmi les* $\overline{cd_x}$ *de* $\overline{X}$ "

$$\equiv$$

$$\neg(\prod_{X} cd_x) \Longleftrightarrow \sum_{\overline{X}} \overline{cd_x} \tag{2.15}$$

$$\Longleftrightarrow \sum_{X} \neg cd_x \tag{2.16}$$

La représentation en RdP (figure 19) est le complémentaire de la figure 17.

3.3.4.3 Il n'existe pas...

Le module M_i évolue, de l'état courant p_i à l'état suivant p_{i+1}, s'il n'existe pas de conditions cd_x vraies parmi les conditions du sous-ensemble X.

Soient $cd_1, cd_2, ..., cd_i, ..., cd_x \in X$, " *Il n'existe pas de* cd_x *vraie dans* X "

$$\equiv$$

" *Toutes les* cd_x *de* X *sont fausses*"

$$\equiv$$

" *Toutes les* $\overline{cd_x}$ *de* $\overline{X}$ *sont vraies* "

$$\equiv$$

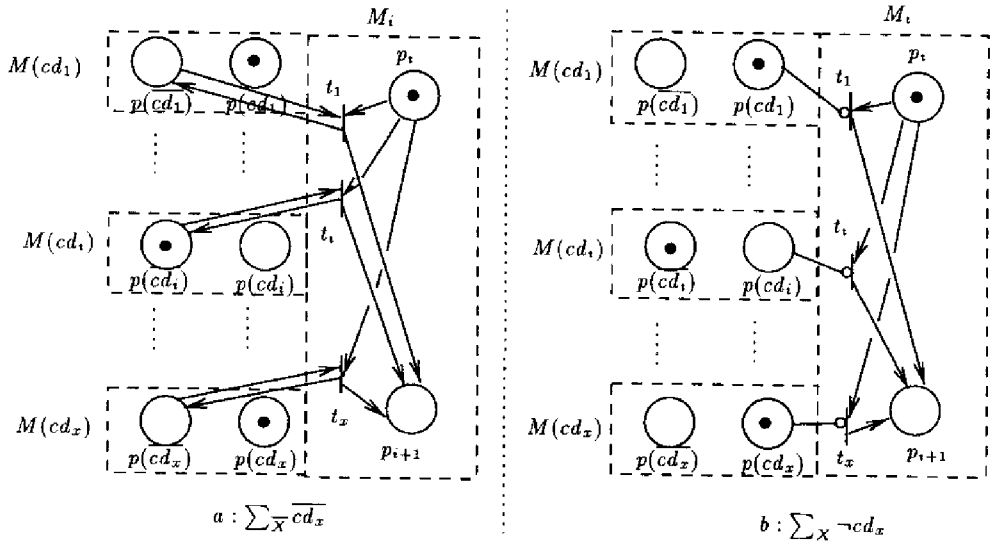

Figure 19: tout... n'est pas vrai

$$\neg(\sum_{X} cd_x) \Longleftrightarrow \prod_{\overline{X}} \overline{cd_x} \tag{2.17}$$

$$\Longleftrightarrow \prod_{X} \neg cd_x \tag{2.18}$$

La représentation est identique à la figure 16.

3.3.5 Procédures algorithmiques élémentaires

3.3.5.1 Tant que... faire

Cette instruction conditionne les évolutions d'un module M_i à la valeur d'une condition : tant que, dans l'état courant p_i, cd_i est fausse, rester dans cet état ; dès que la condition cd_i devient vraie, passer dans l'état suivant p_{i+1}.

"*Tant que cd_i est fausse, faire...* "

La figure 20 représente la modélisation du module M_i qui reste dans l'état p_i tant que la condition cd_i est fausse (tir de t_{tq}), puis passe dans l'état suivant p_{i+1} dès qu'elle est vraie.

3.3.5.2 Si... alors, sinon...

Cette instruction correspond à l'évolution du module M_i, de l'état courant p_i vers les états suivants p_si_{i+1} (tir de t_{si}) ou p_sinon_{i+1} (tir de t_{sinon}), en fonction de la valeur de la condition cd_i. Cette instruction correspond à un choix et est exprimée au moyen de la relation logique OU ($\vee$) (figure 21).

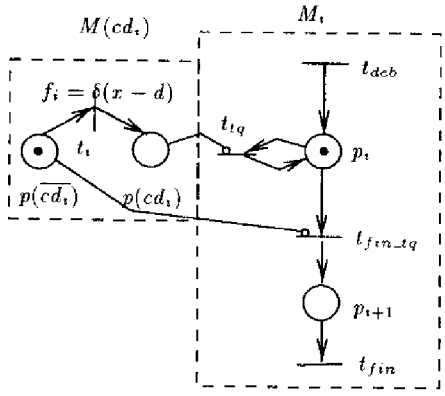

Figure 20: tant que cd_i est fausse, faire...

"*Si cd_i est vraie, alors..., sinon...*"

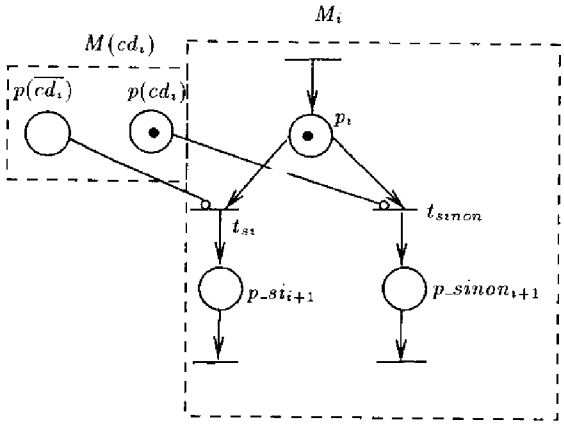

Figure 21: si cd_i alors..., sinon...

3.3.6 Mécanismes élémentaires

Nous allons présenter ci-dessous la modélisation de quelques mécanismes élémentaires qui seront utilisés très couramment dans l'étude de nos systèmes.

3.3.6.1 Cumul de temporisateurs

La représentation de deux temporisations séquentielles nécessite de synchroniser l'activation de la seconde sur la fin de la première.

La figure 22 représente deux temporisations, $tmp1$ et $tmp2$ de durée respective $d1$ et $d2$, $tmp1$ est activée (tir de t_{on1}) par un évènement quelconque et l'activation de $tmp2$ est synchronisée avec la fin de $tmp1$ (tir de $t_{off1/on2}$). De plus, on suppose ici que la temporisation $tmp1$ ne peut être activée que lorsque $tmp2$ n'est pas active.

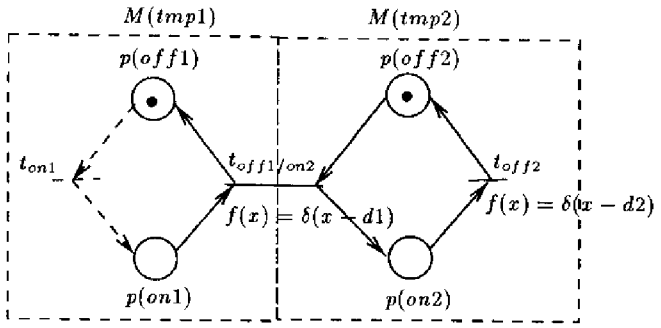

Figure 22: cumul de deux temporisateurs

3.3.6.2 Activation/désactivation d'un délai par une condition

Une temporisation peut être destinée à surveiller les évolutions d'une condition : la temporisation est activée si la condition est fausse et elle est désactivée si elle devient vraie, ce qui implique que la temporisation expire si la condition est restée fausse pendant toute sa durée (dans ce cas, un message peut alors être envoyé à destination d'un autre module).

La figure 23 représente : l'activation de la temporisation du module $M(tmp)$ (tir de t_{on}) si la condition cd_i est fausse ; sa désactivation immédiate (tir de t'_{off}) si cd_i est vraie ; l'expiration du délai (tir de t_{off}) si la condition est restée fausse pendant toute la durée d de la temporisation.

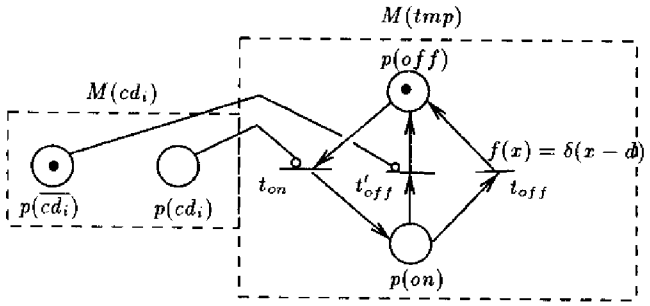

Figure 23: activation/désactivation de tmp par cd_i

3.3.6.3 Déclenchement d'une activité par un délai

Une temporisation peut être activée par un évènement signalant le début d'une activité, cette dernière sera interrompue dès la fin de la temporisation, pour effectuer l'activité suivante.

La figure 24 représente l'activation de l'activité de la place p_i par le tir de t_{i-1} et l'émission synchrone du message $!ev_i$ afin d'activer la temporisation du module $M(tmp)$ de durée d. L'activité de p_i est interrompue, dès la fin de la temporisation, par le test à zéro de la place $M(tmp)[p(on)]$; ceci amène le module M_i dans l'état suivant p_{i+1} par le tir de t_i.

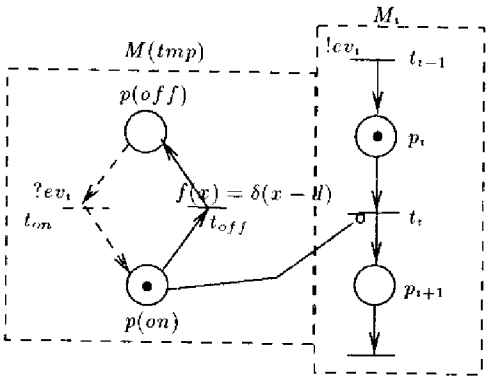

Figure 24: interruption d'une action par la fin d'un délai

4 Analyse

L'intérêt des descriptions formelles est (en plus de leur pouvoir d'expression pour l'activité de spécification/modélisation) leur pouvoir de démonstration propre au formalisme mathématique sous-jacent qui se prête bien à l'activité de validation.

Cette partie présente la méthodologie d'analyse suivie pour effectuer la vérification des spécifications d'un système modélisé en Réseaux de Petri Temporisés Stochastiques en tirant profit de la modularité de la modélisation. En effet, dans le contexte des systèmes complexes temps réel, il est important de pouvoir faire des vérifications *progressives*, tout d'abord du comportement des différents sous-systèmes pris séparément, puis du système complet ; ceci afin de diminuer la complexité des analyses à effectuer. Pour cela, dans un premier temps, nous allons présenter la méthode de construction incrémentale utilisée, puis les différentes étapes ascendantes de construction pour la vérification du modèle global (prenant en compte progressivement l'ensemble des éléments d'un système complexe).

4.1 Constructions partielles

Compte tenu des règles de modélisation proposées, si les modules élémentaires sont bornés, saufs et les places de communication bornées, alors les propriétés générales (graphe borné) sont préservées par les règles de composition choisies. En effet, des études ont été menées [Sou93] [FP94] [SM90] qui prouvent la conservation des propriétés générales (borné, vivacité) d'un modèle construit par composition d'un ensemble de modules élémentaires (également vivants, bornés) par fusion de places, de transitions et par places de communication.

Ainsi, l'intérêt des règles proposées pour la modélisation modulaire et la construction de modules élémentaires est :

- d'avoir des modules élémentaires bornés et autonomes saufs et réinitialisables en inhibant leurs interactions avec les sous-systèmes adjacents et en ne considérant pas leurs places dupliquées (pour la validation complète d'un module, il est alors nécessaire de

pouvoir faire évoluer l'ensemble des marquages des places partagées qu'il utilise). On considère alors que toutes les interactions relatives aux étiquettes et aux places dupliquées sont supprimées afin de permettre toutes les évolutions possibles du comportement du module élémentaire étudié ;

- de permettre d'effectuer des vérifications "partielles" de propriétés spécifiques (choisies par le modélisateur) par la construction du modèle à partir d'un sous-ensemble de modules élémentaires du même sous-système et/ou de sous-systèmes adjacents. Pour cela, il est nécessaire (après avoir déterminé la (les) propriété(s) à vérifier de conserver uniquement les modules élémentaires mis en jeu directement et indirectement pour la vérification de cette (ces) propriété(s). Ceci évite d'avoir à considérer l'ensemble des modèles du système complet.

4.2 Etapes ascendantes de vérification

Les analyses (d'accessibilité) qui peuvent être effectuées sur le graphe d'états probabilisé ont été déjà présentées au chapitre 1 § 4.3.2.

Nous présentons ici la méthodologie de vérification par étapes ascendantes qui utilise le concept de la construction partielle présentée ci-dessus. Pour cela, nous procédons à la vérification des comportements de façon ascendante en tirant profit de la structuration du système considéré en sous-systèmes et de la décomposition de chaque sous-système en un ensemble de modules élémentaires distincts.

La figure 25 illustre les différentes étapes de vérification ascendante explicitées ci-dessous. Chaque étape est caractérisée par deux activités : la modélisation (rectangle hachuré) et la vérification (rectangle gris). Si une phase de vérification n'est pas valide (flèche **N**), la phase précédente de modélisation doit être reprise en modifiant les modèles et/ou les spécifications. Si la phase de vérification est bonne (flèche **O**), l'étape suivante supérieure peut être entreprise.

Les étapes sont :

A : vérifier le comportement d'un module élémentaire après avoir :

1. inhibé ses interactions avec les sous-systèmes adjacents et ses accès en lecture avec les autres modules élémentaires ;
2. connecté un modèle abstrait de son environnement afin de réaliser la fermeture du modèle. Le modèle abstrait de son environnement doit être réduit à la réalisation des seules interactions nécessaires aux évolutions propres du module élémentaire considéré.

Exemple : dans le cas des systèmes de contrôle-commande, vérifier le comportement du module de commande dégradée en le couplant avec un modèle réduit de l'environnement local qui génère les défaillances électriques et reçoit les ordres de commande dégradée.

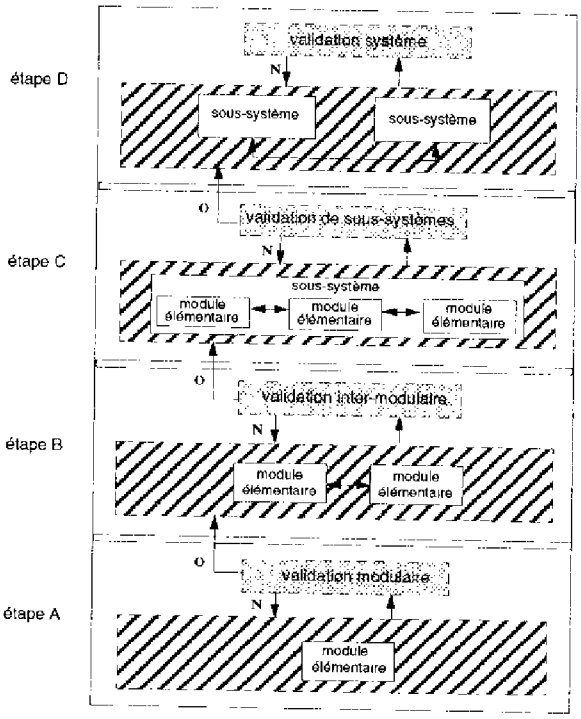

Figure 25: étapes de vérification ascendante

B : vérifier le comportement d'un ensemble de modules élémentaires (appartenant à un même sous-système) après avoir :

1. construit le modèle avec l'ensemble des modules élémentaires conservés par fusion des transitions en rendez-vous immédiat et par fusion des places communes (arcs inhibiteurs) ;
2. inhibé les interactions avec les modules non conservés ;
3. connecté un modèle réduit de l'environnement au modèle.

Exemple : nous avons modélisé et analysé séparément le comportement d'une Entité Application Producteur (ou Consommateur) du réseau de terrain FIP (services application et status de validité temporelle) [JABF92].

C : vérifier le comportement d'un sous-système complet situé sur un site unique en considérant l'ensemble des modules élémentaires qui le constituent après avoir :

1. construit le modèle avec l'ensemble des modules élémentaires le constituant ;
2. connecté un modèle abstrait de l'environnement qui réalise l'ensemble des interactions nécessaires avec au sous-système.

Exemple : nous avons vérifié les principaux services périodiques et mécanismes (status temporels) de FIP des Entités Application de type producteur et consommateur interconnectées par un modèle abstrait de fournisseur de services liaison et fermées par des

modèles simplifiés de l'environnement représentant les comportements nécessaires des Processus Utilisateurs [BJ94].

D : vérifier le comportement du système global en considérant l'ensemble des éléments (sous-systèmes qui le constituent) après avoir :

1. construit le modèle d'un sous-système avec l'ensemble des modules élémentaires par fusion des transitions en rendez-vous immédiat et par fusion des places partagées ;
2. connecté le modèle complet de l'environnement au modèle du système par places de communications.

Notons que, grâce aux concepts présentés de construction modulaire, de composition et vérification partielles ascendantes, il est ici tout à fait envisageable de ne conserver qu'un sous-ensemble des modules élémentaires dans les sous-systèmes locaux impliqués dans la réalisation d'une fonctionnalité que le concepteur veut vérifier et ainsi ne composer, pour la vérification d'une fonction entre sites distants, qu'un sous-ensemble de modules de différents sous-systèmes.

Exemple : nous avons analysé et vérifié les propriétés essentielles d'évitements de défauts multiples du système de contrôle-commande temps réel de EDF réparti sur le réseau de terrain FIP [BJS95b].

L'intérêt de cette méthodologie de vérification est de pouvoir faire la validation progressive du système de façon ascendante par composition partielle des différents modèles du système global qui le composent, et ce jusqu'à la validation du système complet.

5 Conclusion

Trois points essentiels de la méthodologie présentée dans ce chapitre nous paraissent devoir être mis en exergue :

- la **structuration hiérarchique et modulaire**. Cette structuration a permis d'obtenir, à partir d'une vue systémique, une architecture détaillée en terme de blocs fonctionnels de base et de leurs interactions (locales et distantes). Cette structuration a été appliquée, à la fois, aux fonctionnalités de surveillance et de commande d'un système de contrôle-commande, ainsi qu'aux services et mécanismes de communication de réseaux de terrain.

- la **modélisation modulaire et la construction incrémentale de modèles**. La modélisation proposée, au moyen de Réseaux de Petri Temporisés Stochastiques (RdPTS), tire profit de la structuration modulaire de l'architecture (notion de modules élémentaires) et met l'accent sur la criticité de la construction des modèles de modules élémentaires, compte tenu que les transitions des RdPTS ont des attributs temporels ; ceci impose donc des contraintes sur certains types de construction (la fusion de transitions imposent des contraintes – les places de communication n'imposent pas de contraintes).

Notons que, dans les réseaux de Petri classiques où les transitions ne possèdent pas d'attributs temporels, les constructions incrémentales modulaires ne posent pas de problèmes.

Dans ce contexte, trois contributions peuvent être mises en avant :

- définition d'une méthode de construction des modules élémentaires par places dupliquées, basée sur le test à zéro et l'utilisation d'arcs inhibiteurs ; ceci évite la désensibilisation et la réinitialisation des transitions dont les places d'entrées sont accédées en lecture ;
- proposition d'une stratégie hiérarchisée de construction incrémentale de modèle : par fusion de transitions immédiates et de places dupliquées pour la construction incrémentale des modules élémentaires d'un même sous-système (ceci préserve le synchronisme des évolutions internes d'un sous-système) ; par places de communication pour la composition de sous-systèmes adjacents (ceci préserve leur asynchronisme) ;
- définition de modèles d'assertions logiques simples (telles que *et, ou, ou exclusif*) et complexes (telles que *tout est vrai, tout est faux, il existe, tout n'est pas vrai, tout n'est pas faux, il n'existe pas...*). Ces modèles facilitent le passage des descriptions de comportements exprimés en langage naturel (spécifications fonctionnelles), issues des cahiers des charges et/ou des normes, à une modélisation formelle en RdPTS en vue de la validation des spécifications.

- la **vérification par construction partielle.** Grâce aux règles proposées pour la modélisation modulaire et la construction partielle, il est possible d'effectuer des validations progressives : de modules, d'un ensemble de modules, puis de sous-systèmes et enfin, du système global.

Chapitre 3

Présentation des systèmes analysés : l'application de contrôle-commande de EDF et le réseau de terrain FIP

1 Introduction

L'objectif de ce chapitre est de présenter le contexte applicatif de notre travail, c'est-à-dire l'application de contrôle–commande distribuée temps réel que nous considérons et le réseau de terrain FIP.

Ce chapitre comprend deux parties :

- la première partie présente à la fois le système support, à savoir un poste de transformation HTB/HTA (90-63/20 kV) du réseau électrique EDF, et la problématique de ce poste qui justifie la mise en œuvre d'une application de contrôle–commande distribuée temps réel ;
- la deuxième partie présente à la fois les principales caractéristiques du réseau de terrain FIP et les considérations pratiques pour la mise en œuvre de l'application de contrôle-commande sur ce réseau.

2 L'application considérée de contrôle–commande relative à un poste de transformation HTB/HTA

Notre objectif, dans ce paragraphe, est de donner, le plus simplement possible, les éléments essentiels relatifs au fonctionnement d'un poste de transformation HTB/HTA (90-63/20 kV),

de manière à pouvoir appréhender d'une part, la problématique de l'application de contrôle-commande et d'autre part, la nécessité de la répartition sur un réseau de communication.

Soulignons que l'application présentée ci-dessous est une application construite à partir des documents de spécifications du poste de transformation d'énergie électrique HTB/HTA de EDF ([Sab93a], [Sab93b], [AFN], [Sol], [SG93]) dans l'objectif de définir un exemple à la fois significatif pour les besoins de l'étude et proche des problématiques réelles. La description informelle du système et des fonctionnalités a volontairement été simplifiée pour faciliter la compréhension du sujet et pour les besoins de l'étude. Des descriptions plus détaillées peuvent être trouvées dans le rapport [BS95].

2.1 Présentation générale

Un poste de transformation HTB/HTA est un dispositif de base pour la distribution de l'énergie électrique dans le réseau électrique EDF. Ce dispositif (figure 1) est généralement accédé par trois lignes Haute-Tension qui arrivent du réseau de transport et qui l'alimentent en énergie électrique. Plusieurs lignes Moyenne-Tension, appelées lignes départ, partent vers le réseau de distribution pour alimenter les clients.

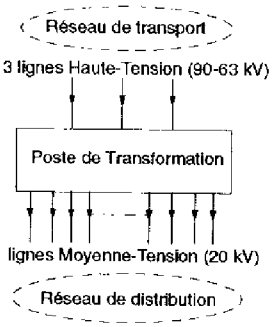

Figure 1: situation d'un poste de transformation HTB/HTA

2.1.1 Fonctions

Outre la fonction première (qui découle de sa définition même), c'est-à-dire la fonction d'abaisser la Haute-Tension (HTB) à la Moyenne-Tension (HTA), le poste de transformation doit traiter les défauts électriques qui apparaissent sur les différents éléments. En conséquence, il doit assurer :

- la sécurité du matériel et des clients : ceci implique des fonctions de détection et d'élimination de défauts électriques (par des coupures de courant sur les lignes en défaut) ;
- la continuité du service pour les clients : dans l'hypothèse où, suite à un défaut électrique, il faut pratiquer des coupures de courant, l'objectif de continuité du service

consiste à minimiser le nombre de clients concernés par les coupures de courant et à ré-alimenter le plus rapidement possible les clients concernés par les coupures.

Des automatismes de contrôle–commande ont été définis afin d'assurer d'une part, la sécurité du matériel et des clients et d'autre part, la continuité du service pour les clients. Ces automatismes mettent en œuvre des techniques de localisation et d'élimination des défauts électriques.

Notre travail concerne précisément l'application définie par ces automatismes de contrôle–commande.

2.1.2 Topologie d'un poste de transformation

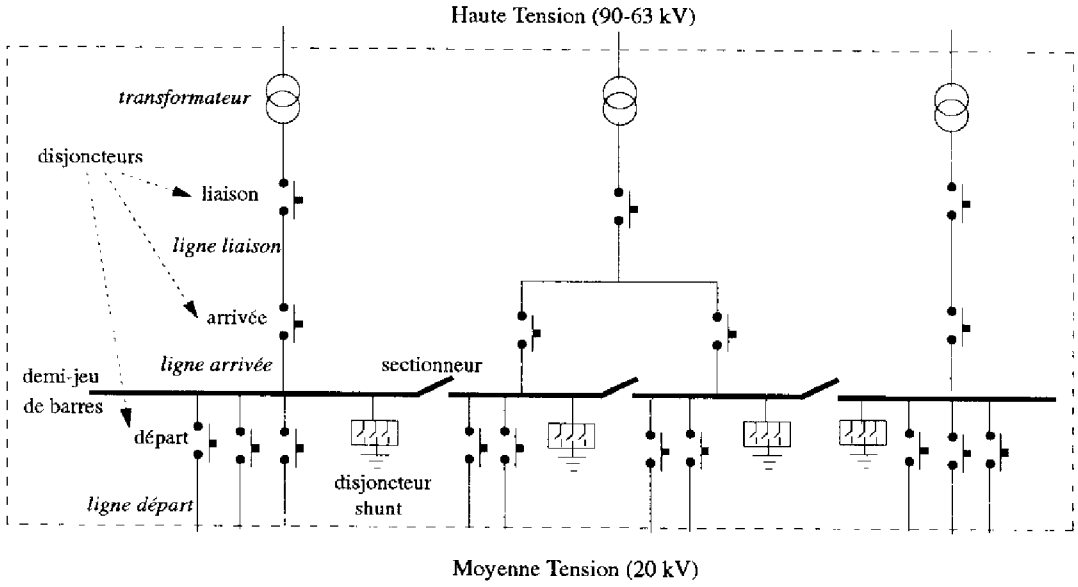

Figure 2: Topologie d'un poste de transformation HTB/HTA

La topologie classique d'un poste de transformation HTB/HTA est représentée sur la figure 2. Elle comprend un ensemble de transformateurs, de disjoncteurs et de sectionneurs de demi-jeu de barres (qui permettent de modifier le schéma d'alimentation des lignes départ) :

- un transformateur par ligne Haute-Tension abaisse la tension ;
- après chaque transformateur, l'électricité est acheminée par une ligne liaison vers un ou plusieurs demi-jeux de barre à travers une ligne arrivée (un demi-jeu de barre est une connexion métallique) ; chaque demi-jeu de barre est connecté à plusieurs lignes départ (le rôle des demi-jeux de barres est de transmettre l'énergie de la ligne liaison vers les lignes départ afin d'alimenter les clients) ; le courant sur les lignes et les demi-jeux de barres est triphasé ;

- un disjoncteur est associé : à la ligne liaison, du côté transformateur (disjoncteur liaison) ; à la ligne arrivée et son demi-jeu de barre (disjoncteur arrivée) ; à chaque ligne départ (disjoncteur départ) ; de plus, des disjoncteurs sont associés à chaque phase des demi-jeux de barre (disjoncteur shunt).

En ce qui concerne les disjoncteurs, on doit en distinguer deux types :

- les disjoncteurs départ, arrivée et liaison qui ont un rôle double (fonctionnel et sécuritaire). En effet, ils permettent de gérer la distribution de l'énergie (un disjoncteur fermé permet la circulation du courant) et sporadiquement de mettre en œuvre des opérations sécuritaires suite à l'apparition de défauts électriques. Les opérations sécuritaires sont basées sur des actions d'ouvertures de disjoncteurs (ce qui annule le courant).

  Notons la hiérarchie des rôles des différents disjoncteurs : le disjoncteur départ n'a qu'un rôle local (ouverture et fermeture d'une ligne départ, ce qui touche les clients de sa ligne) ; le disjoncteur arrivée a un rayon d'action plus étendu (son ouverture et sa fermeture affectent tous les clients connectés au demi-jeu de barres alimenté par cette arrivée) ; le disjoncteur liaison a un rayon d'action global (permettre l'alimentation ou la non alimentation de tous les clients concernés par une ligne Haute-Tension).

- les disjoncteurs shunt qui ont uniquement un rôle sécuritaire. Un disjoncteur shunt est constitué de trois disjoncteurs, un sur chaque phase du courant appelée pôle. Ils interviennent, généralement, avant les disjoncteurs départ, arrivée et liaison en se fermant indépendamment sur chacune de ces phases, ce qui a pour but de court-circuiter le courant.

L'ensemble des éléments d'un poste de transformation, qui sont alimentés à travers la même source d'énergie (transformateur), définissent une partie du poste dite en continuité électrique.

2.1.3 Sur les défauts électriques dans une partie en continuité électrique

2.1.3.1 Définitions et propriétés des défauts électriques

Un défaut électrique est une anomalie électrique qui peut apparaître (naître) sur une phase (défaut monophasé) ou sur plusieurs phases (défaut polyphasé) d'une ligne (départ, arrivée, liaison) ou d'un demi-jeu de barres.

Un défaut électrique est caractérisé par : une modification de la valeur du courant (sur une ligne) et/ou de la tension (entre le demi-jeu de barres et le neutre) par rapport à une valeur nominale (franchissement d'un seuil) et la durée de persistance de cette perturbation.

Les défauts électriques peuvent apparaître sur : les lignes départ, les demi-jeux de barres, les lignes arrivée et/ou les lignes liaison (cf. figure 3).

Une propriété importante d'un défaut électrique est la suivante : un défaut, qui apparaît à un endroit est également vu en amont de cet endroit ; ceci est dû à des phénomènes de propagation amont des défauts.

2.1.3.2 Le contrôle des défauts électriques

La mise en œuvre

Afin d'effectuer le contrôle des différents disjoncteurs, des éléments sont introduits dans le poste de transformation HTB/HTA. Il s'agit d'une part, des capteurs et des protections numériques et d'autre part, des cellules d'automatismes :

- capteurs et "protections numériques" : les capteurs (respectivement départ, shunt, arrivée, liaison) sont connectés sur les différentes lignes (respectivement départ, demi-jeu de barres, arrivée, liaison) et ils effectuent la mesure du signal électrique sur leur ligne respective. Cette mesure est restituée aux protections qui sont des équipements effectuant un traitement du signal. Les protections numériques réalisent alors la signalisation des défauts électriques aux cellules. Si un défaut électrique est présent et persiste pendant un certain temps, appelé `Temps de Pré-confirmation` (temps nécessaire à la protection pour effectuer le traitement du signal), le défaut est dit `pré-confirmé`. Les temps de pré-confirmation varient en fonction de la protection. On appelle : T_{pc} ce délai au niveau du shunt (de l'ordre de $50ms$) et T_{PC} ce délai au niveau des départs, arrivées et liaisons (de l'ordre de $100ms$). Afin de simplifier, nous considérons qu'un ensemble capteur/protection constitue un *capteur intelligent* ;

- cellules : des cellules d'automatismes (systèmes de contrôle) sont associées à ces *capteurs intelligents* (on a des cellules départ, shunt, arrivée et liaison). Chaque cellule est informée, par le *capteur* qui lui est associé, de l'existence d'un défaut pré-confirmé et, suite à cette information, la cellule va alors programmer des actions de traitement de défauts qui consistent en des commandes d'ouverture et/ou de fermeture sur son disjoncteur associé.

  Notons que la cellule shunt ne peut actionner son disjoncteur shunt que dans le cas d'un défaut monophasé. Dans le cas d'un défaut polyphasé, le fonctionnement du disjoncteur shunt est inhibé, c'est-à-dire que la cellule shunt ne peut plus exercer de commande sur lui pendant un certain temps (2s) : on dit que le shunt est verrouillé.

La problématique du contrôle

On peut appréhender cette problématique d'une part, en considérant les différents endroits d'apparition des défauts électriques et d'autre part, en intégrant les conséquences du phénomène de propagation amont sur la présence de défauts pré-confirmés (figure 3). On peut voir sur cette figure que, si un défaut apparaît à un endroit, il est également pré-confirmé dans les cellules amonts ; toutes les cellules concernées (cellule proche du lieu d'apparition et cellules amonts) vont entamer leur procédure de traitement de défauts.

Il est donc essentiel, dans ce contexte, que ces traitements soient coordonnés en tenant compte de la hiérarchie des fonctions de surveillance répartie associées aux disjoncteurs, à savoir les disjoncteurs départ, arrivée et liaison. Plus précisément, il faut que les cellules les plus proches du lieu d'apparition du défaut puissent réaliser leur traitement spécifique

défaut préconfirmé / endroit d'apparition du défaut	défaut shunt préconfirmé	défaut départ préconfirmé	défaut arrivée préconfirmé	défaut liaison préconfirmé
ligne départ	X	X	X	X
ligne arrivée et/ou demi-jeu de barres	///	///	X	X
ligne liaison	///	///	///	X

Figure 3: lien entre endroits d'apparition des défauts électriques et défauts pré-confirmés

(traitement qui ne peut avoir que des conséquences locales par rapport à l'origine du défaut) avant qu'une cellule en amont ne réalise le sien (qui forcément a des conséquences plus larges)... et ainsi de suite... : on parle de techniques de localisation des défauts.

Les défauts qui apparaissent sur les lignes départs sont des défauts dont le contrôle est d'une part, d'une importance capitale car ces défauts se situent au plus près des clients et d'autre part, le plus difficile à mettre en œuvre car le phénomène de propagation amont est alors maximal.

Dans le cadre de l'étude présentée, nous ne considérons que des défauts qui apparaissent sur les lignes départs (défauts départ).

2.1.4 Les techniques de localisation et de traitement des défauts départ

2.1.4.1 Localisation des défauts

Elle est basée sur des principes appelés principes de sélectivité. Deux principes de sélectivité sont utilisés : la sélectivité temporelle et la sélectivité logique.

La sélectivité temporelle est basée sur le principe des différences temporelles de traitement entre les cellules départ, arrivée et liaison, c'est-à-dire : si un défaut apparaît sur une ligne départ, le temps est laissé à sa cellule départ d'exercer son traitement avant que, si le traitement n'aboutit pas, celui de la cellule arrivée (qui l'alimente) intervienne... et ainsi de suite. Notons que les cellules fonctionnent indépendemment du point de vue logique ; aucune cellule n'a connaissance de l'état des autres cellules (chaque cellule fonctionne de manière "aveugle").

La sélectivité logique conserve les principes premiers de la sélectivité temporelle (différences temporelles de traitement entre les cellules départ, arrivée et liaison) et introduit des notions de coopérations entre les différentes cellules (les cellules ne travaillent plus "en aveugle" comme dans le cas de la sélectivité temporelle mais échangent leurs états) : ceci doit permettre d'améliorer les résultats des mécanismes de traitement des défauts.

2.1.4.2 Traitement des défauts

Considérons un défaut qui donne des défauts pré-confirmés shunt, départ, arrivée et liaison. Le traitement comprend deux composantes :

- tout d'abord, des essais d'élimination de défauts sont commandés successivement par la cellule shunt et la cellule départ (le défaut shunt étant le premier pré-confirmé car $T_{pc} = 50ms$). On a donc en premier les essais de la cellule shunt ; si ces essais n'aboutissent pas, le défaut départ va être pré-confirmé ($T_{PC} = 100ms$) et donc la cellule départ va entreprendre des essais. Ces essais consistent en des actions d'ouverture et/ou de fermeture sur les disjoncteurs shunt et départ que nous présentons ci-après.

  Remarque : nous avons considéré implicitement un défaut monophasé ; si on avait un défaut polyphasé, il n'y aurait pas la phase d'essai d'élimination commandée par la cellule shunt et il y aurait uniquement la phase contrôlée par la cellule départ.

- si les essais d'élimination des défauts, commandés par les cellules shunt et départ, n'aboutissent pas, des coupures de service contrôlées par les cellules arrivée et/ou liaison peuvent intervenir ; ces coupures sont obtenues au moyen d'ouvertures définitives des disjoncteurs arrivée et/ou liaison.

Nous présentons maintenant les caractéristiques essentielles des actions commandées par les cellules shunt, départ, arrivée et liaison.

Essais d'élimination

Au niveau du disjoncteur shunt

Quand la cellule shunt est informée de l'existence d'un défaut pré-confirmé sur une seule phase (défaut monophasé) du demi-jeu de barres, elle agit sur le disjoncteur shunt en deux étapes : 1) en fermant le pôle relatif à la phase en défaut pendant 250ms (ce qui annule la tension sur la phase concernée) ; 2) en ré-ouvrant définitivement ce pôle au bout des $250ms$.

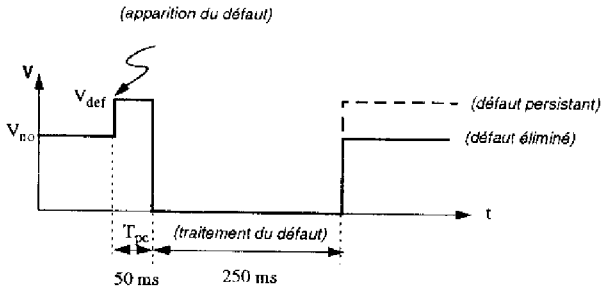

Figure 4: tension de défaut au niveau du shunt

La figure 4 représente l'évolution de la tension V de la phase en défaut où V_{no} et V_{def} sont respectivement, la valeur de la tension nominale et la valeur de la tension de défaut de la phase considérée. Après la durée (50ms) de pré-confirmation du défaut, la fermeture du disjoncteur shunt est effectuée (250ms) et on peut alors avoir les situations suivantes :

- soit le défaut a disparu, cas où la tension reprend sa valeur nominale (trait noir plein) ;
- soit le défaut persiste, cas où la tension reste à la valeur de défaut (trait pointillé).

Au niveau du disjoncteur départ

Quand une cellule départ est informée de l'existence d'un défaut départ pré-confirmé (notons que ceci signifie, dans l'hypothèse d'un défaut monophasé, que la fermeture du disjoncteur shunt n'a pas éliminé le défaut), elle a le comportement suivant :

- tout d'abord, la cellule surveille si le défaut persiste pendant un temps prédéterminé, fixe appelé **Temps de Confirmation** (noté TD_{cnf}). Un défaut départ pré-confirmé qui persiste pendant le temps TD_{cnf} est appelé un défaut départ confirmé. S'il n'existe pas de défaut départ confirmé (c'est-à-dire que le défaut pré-confirmé a disparu avant la fin du délai de confirmation TD_{cnf}), la cellule n'entreprend pas de traitement ;
- ensuite, si un défaut départ a été confirmé, la cellule commence le traitement qui comprend, au niveau de son disjoncteur départ, trois cycles de réenclenchement successifs (un cycle rapide suivi de deux cycles lents).

  Un cycle de réenclenchement comprend : a) une ouverture du disjoncteur départ pendant une durée fixe, appelée Temps d'Ouverture (notée TO) ; b) une fermeture de durée fixe, appelée Temps de Fermeture (notée TF).

  Si, à la fin du temps de fermeture, un défaut (pré-confirmé) est toujours présent, le cycle suivant est demandé et ainsi de suite... Il existe trois cycles de réenclenchement successifs (rapide → lent_1 → lent_2) dont les durées d'Ouverture sont croissantes. Si, à la fin du cycle rapide, le défaut est toujours présent, le cycle lent_1 est démarré ; si, à la fin du cycle lent_1, le défaut est toujours présent, le cycle lent_2 est démarré ; si, à la fin du cycle lent_2, le défaut est toujours présent, l'ouverture du disjoncteur est définitive (coupure du service) en **attendant l'intervention de l'opérateur.**

  Dans le cas contraire, si aucun défaut pré-confirmé n'est présent à la fin du temps de fermeture d'un cycle, ceci signifie que le cycle a permis l'élimination du défaut.

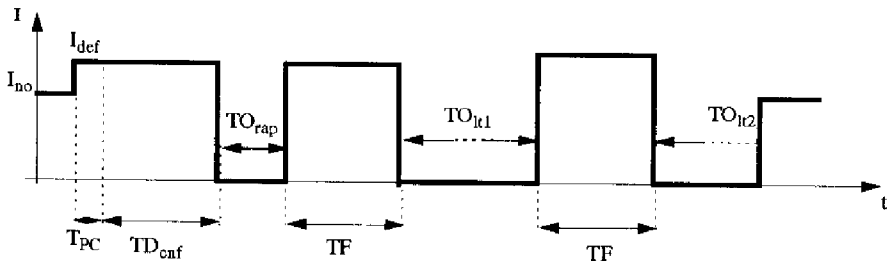

Figure 5: défaut éliminé après trois cycles de réenclenchement départ

Le chronogramme temporel des cycles de réenclenchement est représenté sur la figure 5 ; les temps d'ouverture associés respectivement aux cycles rapide, lent_1 et lent_2 sont

tels que $TO_{rap} \ll TO_{lt1} \ll TO_{lt2}$. Notons que pendant le temps d'ouverture du disjoncteur, le défaut électrique sur la ligne est supprimé (puisque le courant est annulé). La fermeture du disjoncteur permet, en rétablissant le courant, de voir si le défaut a réellement disparu ou non (c'est-à-dire si un défaut pré-confirmé existe toujours à la fin de TF).

Coupures de services : au niveau du disjoncteur arrivée et du disjoncteur liaison

Au niveau du disjoncteur arrivée, après que la cellule arrivée a été informée, par la protection arrivée, de l'existence d'un défaut arrivée pré-confirmé, la cellule arrivée attend que ce défaut persiste pendant un temps, appelé temps de confirmation arrivée TA_{cnf}, avant de commander l'ouverture définitive du disjoncteur arrivée (seule une manoeuvre de l'opérateurpeut ramener le disjoncteur fermé).

La même logique de traitement est appliquée au niveau du disjoncteur liaison ; on note TL_{cnf} le temps de confirmation liaison et on a $TL_{cnf} > TA_{cnf}$.

2.2 Sur les principes de la sélectivité temporelle et de la sélectivité logique

Dans ce paragraphe, nous montrons les mécanismes qui permettent de mettre en œuvre ces principes. Tout d'abord, nous présentons la sélectivité temporelle en faisant apparaître, en particulier, ses insuffisances, et ensuite, nous montrons l'intérêt de la sélectivité logique pour corriger ces insuffisances.

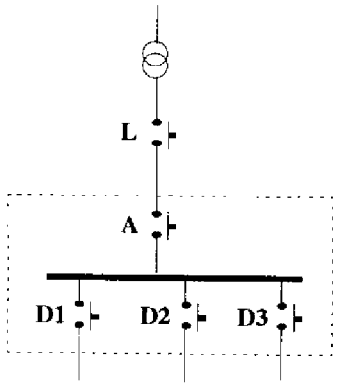

Figure 6: éléments en continuité électrique considérés

Afin de faire cette présentation et dans toute la suite de l'étude, nous considérons comme exemple la partie en continuité électrique d'un poste HTB/HTA qui est sur la figure 6 et qui comprend : trois lignes départ (D1, D2 et D3) et leur disjoncteur départ respectif qui les relie à un demi-jeu de barres (on ne considère pas le disjoncteur shunt) ; une ligne arrivée et son disjoncteur arrivée (A), une ligne liaison et son disjoncteur liaison (L). De plus, l'exemple se limite à la sous-partie constituée des départs et de l'arrivée (cadre en pointillé).

2.2.1 Sélectivité temporelle

Les spécifications des différentes temporisations dans les cellules sont les suivantes :
- cellule départ : $TD_{cnf} = 500ms, TO_{rap} = 300ms, TO_{lt1} = 15s, TO_{lt2} = 30s, TF = 500ms$,
- cellule arrivée : $TA_{cnf} = 2.TD_{cnf} = 1s$,
- cellule liaison : $TL_{cnf} = 3.TD_{cnf} = 1.5s$.

Nous considérons deux cas : 1/ un défaut apparaît sur la ligne départ D1 (ce cas 1 montre le principe même de la sélectivité temporelle) ; 2/ deux défauts apparaissent successivement sur la ligne départ D1, puis sur la ligne départ D2 (ce cas 2 montre l'insuffisance de cette technique de sélectivité).

2.2.1.1 Cas 1

On appelle I_{D1} le courant sur la ligne départ $D1$ et I_A le courant sur la ligne arrivée. Nous présentons sur la figure 7 les diagrammes temporels des courants I_{D1} et I_A en mettant en évidence la durée du défaut et les temporisations qui interviennent.

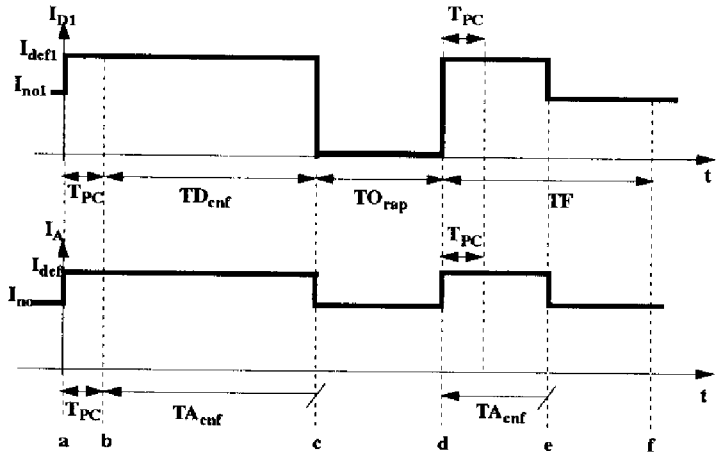

Figure 7: scénario temporel du cas 1

- Le défaut est représenté au niveau du départ $D1$ par la valeur I_{def1} (courant de défaut départ 1) au lieu de la valeur I_{no1} (courant nominal). Lorsqu'un défaut apparaît à l'instant **a**, il est pré-confirmé à l'instant **b** (au bout de T_{PC}) s'il n'a pas disparu. La temporisation de confirmation TD_{cnf} est alors activée (cette activation est notée $\leftarrow TD_{cnf}$). Nous supposons que le défaut persiste jusqu'à la fin de la temporisation de confirmation (notée $TD_{cnf} \rightarrow$) à l'instant **c**, ce qui déclenche l'exécution du cycle de réenclenchement rapide. Ce cycle comprend : tout d'abord, l'ouverture du disjoncteur départ D1 pendant le temps TO_{rap}, ce qui annule le courant I_{D1} et entraîne donc la disparition du défaut momentanément ; ensuite, la fermeture du disjoncteur D1 à l'instant **d** de durée fixe TF. On suppose que le défaut disparaît à l'instant **e**, avant la fin du temps TF à l'instant **f** (notons que si le défaut pré-confirmé avait été présent à la fin du temps TF, on aurait démarré le cycle de réenclenchement lent_1).

– Le défaut est aussi observé au niveau de l'arrivée (propriété de propagation amont des défauts) à travers la valeur I_{def} (courant de défaut de l'arrivée A) au lieu de la valeur nominale I_{no} (courant normal). Le défaut apparaît sur A au même instant **a** et est également pré-confirmé à l'instant **b**. Ceci active alors le délai de confirmation TA_{cnf} de la cellule arrivée. Cette temporisation est désactivée dès que le défaut disparaît à l'instant **c** (dû à l'ouverture du disjoncteur D1). La fermeture automatique de D1 (entre les instants **d** et **e**) fait réapparaître le défaut pré-confirmé et déclenche l'activation de TA_{cnf} qui est ensuite désactivée à la disparition du défaut (instant **e**), avant d'être arrivée à son terme.

Cet exemple simple montre bien le principe de la sélectivité temporelle : l'action sur le disjoncteur départ qui arrive à annuler le défaut ; la temporisation arrivée TA_{cnf} qui est activée mais n'arrive pas à son terme (notons que même si le défaut n'était pas annulé avant la fin de la durée de fermeture TF du disjoncteur départ, on aurait eu les cycles lent_1 et lent_2, mais la temporisation ne serait pas arrivée à son terme car $TA_{cnf} \gg TF$). La cellule arrivée n'effectue donc pas de traitement car la cellule départ (en aval) a réussi à traiter le défaut.

2.2.1.2 Cas 2

Nous représentons (figure 8) l'occurrence de deux défauts départ (sur les lignes D1 et D2) qui se chevauchent dans le domaine temporel.

– Les défauts sur les lignes départ D1 et D2 sont toujours représentés par les valeurs I_{def1} et I_{def2} au lieu respectivement, des valeurs nominales I_{no1} et I_{no2}. Les défauts apparaissent sur les lignes D1 et D2, respectivement aux instants **a** et **c** (dans l'exemple, on considère que c=a+300ms) et sont pré-confirmés à leur cellule respective aux instants **b** et **d** ; les temporisations TD_{cnf} sont alors activées. Nous supposons que les défauts persistent jusqu'à la fin des temporisations de confirmation, ce qui provoque l'ouverture en cycle rapide des cellules départ D1 et D2, respectivement, aux instants **e** et **f** (l'instant **f** est choisi délibérément pour correspondre à la fin de TO_{rap} de D1 et à la fin de TD_{cnf} de D2, ce qui résulte du choix de l'instant d'occurrence du défaut sur D2). Au bout de la durée fixe des délais d'ouverture rapide TO_{rap}, respectivement aux instants **f** et **i**, la fermeture de durée TF doit être effectuée. On suppose que le défaut sur D1 est encore présent pendant toute la durée de cette fermeture ; une supposition quelconque sur la durée du défaut D2 ne change rien à l'analyse de ce scénario (comme nous allons le voir ci-dessous).
- Un défaut apparaît également à l'instant **a** sur la ligne arrivée A (dû au défaut sur D1) et est pré-confirmé à sa cellule à l'instant **b** et alors, la temporisation de confirmation TA_{cnf} est activée. La ligne arrivée reste en défaut depuis l'instant **b** jusqu'à l'instant **h** (où la temporisation TA_{cnf} arrive alors à son terme), ceci sans connaître l'origine de son défaut (par principe même).
La persistance de ce défaut sur A est due, entre les instants : **a** et **c**, au défaut sur D1 ; **c** et **e**, aux défauts simultanés sur D1 et D2 ; **e** et **f**, au défaut sur D2 ; **f** et **h**, au défaut sur D1.
Quand la temporisation TA_{cnf} expire à l'instant **h**, le disjoncteur arrivée est alors ouvert, ce

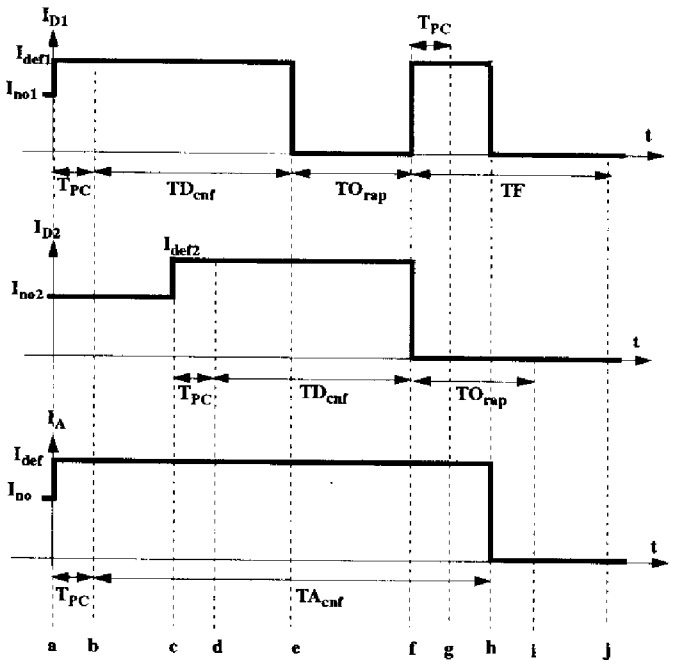

Figure 8: scénario temporel du cas 2

qui annule le courant dans toutes les lignes départ alimentées par A. On voit en particulier, que quelle que soit l'hypothèse que l'on peut faire sur la durée de persistance du défaut sur D2 après son délai d'ouverture TO_{rap}, le courant I_{D2} restera nul à cause de l'ouverture définitive du disjoncteur arrivée.

On voit donc que certaines configurations d'apparition de défauts peuvent provoquer l'ouverture *prématurée* du disjoncteur arrivée, alors que les traitements d'élimination des défauts par les disjoncteurs départ ne sont pas totalement terminés (ici les cycles rapides n'ont pas pu se terminer au niveau des cellules départ D1 et D2). Ceci est dû à l'accumulation temporelle des défauts départ qui fait apparaître un défaut arrivée de manière continue et amène à l'expiration de TA_{cnf}.

Cet exemple montre l'insuffisance de la technique de sélectivité temporelle à deux niveaux :

- l'apparition de défauts multiples, c'est-à-dire qui résultent du chevauchement temporel de défauts sur plusieurs départs distincts (ici entre les instants **c-e**). Or, ces défauts sont très dangereux pour la sécurité du matériel et des hommes : la sécurité n'est donc pas assurée ;

- l'ouverture du disjoncteur arrivée a une influence pénalisante sur les clients de la ligne départ D3 qui sont privés de courant alors que leur ligne n'était pas en défaut (cf. figure 6) : la continuité de service n'est pas assurée.

La sélectivité logique a été introduite afin de pallier principalement à ces deux insuffisances de garantie de fonctionnement.

2.2.2 Sélectivité logique

2.2.2.1 Coopérations de base

La sélectivité logique définit des coopérations entre les différentes cellules réparties. Quatre niveaux de coopération sont introduits :

1. shunt–départ
2. départ–départ
3. départ–arrivée
4. arrivée–liaison

La coopération 1 a pour but, dans le cas où un défaut pré-confirmé est apparu au niveau d'une cellule départ x (qui active sa phase de confirmation) de raccourcir la phase de confirmation pour déclencher au plus tôt l'ouverture du cycle de réenclenchement rapide grâce à l'information reçue du shunt (*autorisation d'ouverture rapide*). Le shunt a le comportement suivant :

- s'il voit un défaut monophasé pré-confirmé, la cellule shunt envoie, 100 ms après la fermeture du pôle concerné par le défaut (cf. figure 4), un message appelé *autorisation d'ouverture rapide* ; ce message sera utilisé par les cellules départ, lorsqu'elles le reçoivent au cours de leur phase de confirmation de défaut (ceci signifie que l'action d'essai d'élimination du défaut par le shunt n'a pas donné de résultat) ;
- s'il voit un défaut polyphasé pré-confirmé, la cellule shunt envoie immédiatement le message *autorisation d'ouverture rapide*. Notons que dans ce cas, ce message sera forcément utilisé par les cellules départ puisque le shunt ne peut pas effectuer d'actions d'élimination sur des défauts polyphasés.

La coopération 2 a pour but d'éviter et éventuellement de raccourcir les durées d'existence des défauts doubles. Elle est basée sur les échanges suivants : toute cellule départ, à l'instant où elle constate l'apparition d'un défaut départ pré-confirmé ou la disparition d'un défaut départ, informe les autres cellules départ de ces évènements (*défaut départ*, *pas de défaut départ*). La mise en œuvre de cette coopération est faite à travers deux règles :

- règle 2.a : si une cellule départ x connaît l'existence d'un autre défaut départ à l'instant t de manifestation d'un défaut départ x pré-confirmé sur sa ligne, elle a le comportement suivant :
  1. si cet instant t est le début de sa phase de confirmation (c'est-à-dire avant la fin de sa phase de pré-confirmation), elle n'exécute pas la phase de confirmation et passe immédiatement dans la phase d'ouverture du cycle de réenclenchement rapide (TO_{rap}). Le défaut départ x est donc supprimé par cette phase d'ouverture pendant TO_{rap} (donc on n'a pas de défaut double) ;
  2. si cet instant t est au cours d'une phase de fermeture d'un cycle de réenclenchement, on passe immédiatement dans la phase d'ouverture du cycle de réenclenchement suivant. Le défaut départ x est donc supprimé immédiatement par cette phase d'ouverture et donc également le défaut double.

- règle 2.b : si une cellule départ x connaît à la fin normale de la temporisation d'ouverture d'un cycle de réenclenchement, l'existence d'un autre défaut départ, elle retarde la fermeture de son disjoncteur x jusqu'à ce qu'elle soit informée de la disparition de l'autre défaut départ. On dit que la cellule départ x attend "l'autorisation des autres cellules départ" (c'est-à-dire qu'il n'existe plus d'autres départs en défaut). En effet, dans l'hypothèse où en refermant le disjoncteur x, il y aurait un défaut départ x toujours présent, on aurait donc création d'un défaut double. Cette durée d'attente est limitée à 2s (après quoi, le disjoncteur sera laissé définitivement ouvert en attendant l'intervention de l'opérateur).

La coopération 3 a pour but, en particulier, d'éviter l'ouverture prématurée du disjoncteur arrivée avant que les disjoncteurs départ aient pu effectuer complètement leurs cycles de réenclenchement. L'évitement de cette ouverture prématurée est obtenu par l'envoi, par les cellules départ, du message *défaut départ* à la cellule arrivée (celle-ci est informée du traitement du défaut départ par la cellule départ). La mise en œuvre de cette coopération se fait à travers la règle suivante :

- règle 3 : lorsque la cellule arrivée, en cours de confirmation, reçoit un message de défaut départ x et si sa temporisation TA_{cnf} est démarrée (depuis un temps supérieur à une valeur d très petite)[1], alors la temporisation de confirmation arrivée est ré-initialisée (relancée) une fois par départ distinct et au plus trois fois ; ceci permet de retarder l'expiration de TA_{cnf}.

La coopération 4 a pour but, en particulier, d'éviter l'ouverture prématurée du disjoncteur liaison. Lorsqu'une cellule arrivée ré-initialise sa temporisation de confirmation de défaut, elle envoie, à la cellule liaison qui l'alimente, un message de relance afin de ré-initialiser également la temporisation de confirmation de la liaison[2].

Notons cependant, que les coopérations 2, 3 et 4, avec d'autres messages, permettent également de gérer les conséquences de pannes des éléments (capteurs, disjoncteurs...). Elles concernent `le déclenchement, par l'arrivée, d'un départ` dont les protections sont défaillantes et le `déclenchement de l'arrivée en secours d'un départ` dont le disjoncteur est défaillant. Le détail de ces échanges est décrit dans le rapport [BS95]. Dans le cadre du travail présenté dans ce mémoire, nous considérons uniquement les coopérations 1, 2 et 3 (la partie présentée).

2.2.2.2 Sur la pertinence de la sélectivité logique

Nous reprenons le cas 2 du scénario de la sélectivité temporelle traité en intégrant les mécanismes de la sélectivité logique (le scénario nous permet de visualiser les mécanismes des coopérations 2 et 3).

[1] Ceci a été introduit ici pour pallier à une incomplétude des spécifications en ce qui concerne les retards liés au transfert sur le réseau pour le déclenchement simultané des temporisations départ et arrivée.

[2] Cette coopération est présentée ici à titre indicatif, mais elle ne sera pas utilisée par la suite.

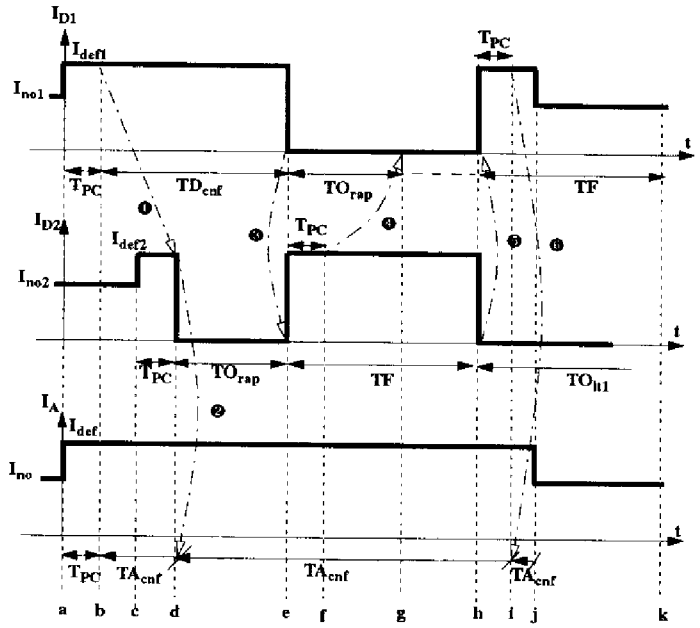

Figure 9: scénario temporel avec la sélectivité logique

Ce scénario est représenté sur la figure 9 (nous supposons que le défaut D2 est présent pendant toute la durée de fermeture TF de son cycle de réenclenchement rapide). Ce scénario est toujours représenté en termes de diagrammes temporels des courants I_{D1}, I_{D2}, I_A. Nous supposons que les échanges entre les cellules sont instantanés, afin de réaliser des changements d'états synchrones. Les flèches en pointillé indiquent les échanges des cellules émettrices vers les cellules réceptrices et qui provoquent des évolutions ou changements d'état au niveau de ces dernières.

– Lorsque le premier défaut apparaît sur le départ D1 à l'instant **a**, il apparaît également au même instant sur la ligne arrivée A. Au bout du temps de pré-confirmation T_{PC}, le défaut étant toujours présent, les temporisations de confirmation, respectivement TD_{cnf} du départ D1 et TA_{cnf} de l'arrivée A, sont activées à l'instant **b**. A cet instant, la cellule départ D1 diffuse l'information de défaut D1 à tout le monde (coopération 2), mais cette information ne provoque pas la réinitialisation de TA_{cnf} (puisque nous supposons les échanges instantanés et donc de durée inférieure à la valeur d) (cf paragraphe 2.2.2.1).

– Lorsque le deuxième défaut apparaît sur le départ D2 à l'instant **c**, ce défaut est pré-confirmé au bout de T_{PC} à l'instant **d**. La cellule D2 diffuse donc vers les autres cellules l'information relative à l'apparition de son défaut (coopération 2), ce qui a pour effet de ré-initialiser (flèche en pointillé 2) la temporisation TA_{cnf} de l'arrivée (coopération 3). Ensuite, la cellule D2 ayant eu connaissance de l'existence du défaut sur D1 (représentée par la flèche en pointillé 1), elle exécute immédiatement l'ouverture du cycle de réenclenchement rapide (règle 2.a de la coopération 2) et active TO_{rap} ; ceci entraîne (étant donné que le disjoncteur départ est ouvert) la disparition du défaut, ce qui va être diffusé aux autres cellules.

– Lorsque la temporisation de confirmation normale de D1 arrive à terme à l'instant **e**, l'ouverture rapide de D1 est effectuée. Son défaut disparaît et elle en informe les autres cellules. Ceci provoque l'activation de la phase de fermeture de la cellule D2 (flèche en pointillé 3), qui correspond (dans ce cas) à l'instant de fin normale de la temporisation d'ouverture TO_{rap} de D2.

– Au bout du délai d'ouverture TO_{rap} de D1 à l'instant **g**, cette phase d'ouverture est prolongée car la cellule D1 a eu connaissance d'un défaut sur D2 (flèche en pointillé 4).

– A la fin du délai de fermeture TF de D2 à l'instant **h**, le départ D2 s'ouvre en cycle de réenclenchement suivant (lent_1) car un défaut local pré-confirmé est toujours présent et il informe les autres cellules de la disparition de son défaut (puisqu'il vient d'ouvrir le disjoncteur). La cellule D1 effectue alors sa fermeture à l'instant **h** car elle a connaissance de la disparition du défaut en D2 (flèche en pointillé 5).

– Lorsque le défaut D1 est pré-confirmé au bout de T_{PC} à l'instant **i**, la cellule D1 informe les autres cellules, ce qui provoque la réinitialisation de la temporisation de confirmation TA_{cnf} de l'arrivée (flèche en pointillé 6) à l'instant **i** (car elle n'a pas encore été ré-initialisée par D1).

– Lorsque le défaut D1 disparaît définitivement à l'instant **j**, sa disparition est également observée au même instant sur la ligne de l'arrivée et sa temporisation de confirmation est désactivée.

Cet exemple met donc en évidence que :

- il n'y a pas de défaut double,
- la continuité de service (en particulier pour D3) est assurée ; l'arrivée n'est pas ouverte car elle est relancée grâce à la connaissance des défauts départ. Ainsi, le temps est laissé aux départs D1 et D2 afin de traiter leur défaut par leurs cycles de réenclenchement.

2.2.2.3 Remarque

Nous avons considéré pour les scénarios analysés dans le cadre de la sélectivité logique la valeur $TD_{cnf} = 500\,ms$ qui est la même valeur que celle utilisée avec la sélectivité temporelle. Notre intention était de montrer l'influence de la coopération entre les cellules. Notons qu'avec la sélectivité logique, la temporisation TD_{cnf} a une nouvelle valeur de $300\,ms$, ce que nous considérerons par la suite.

2.3 Sur la mise en œuvre de la sélectivité logique : une application de contrôle–commande distribuée

2.3.1 La problématique

Le contrôle des défauts électriques dans un poste de transformation, basé sur les principes de la sélectivité logique, est une application répartie de contrôle de procédés temps réel. Elle peut être représentée de façon abstraite comme une boucle de contrôle (*feedback loop*) (figure 10) où : les signaux électriques issus du poste de transformation constituent le procédé à contrôler ; les capteurs (capteurs intelligents) constituent l'interface entre le procédé et le système de contrôle (ils informent le contrôleur de l'*état courant* du procédé) ; les cellules constituent le système de contrôle qui ordonne les traitements afin d'éliminer les défauts ; les disjoncteurs constituent les actionneurs à travers lesquels le contrôleur peut agir sur le procédé (si l'*état attendu* ne correspond pas à l'*état courant du procédé*). Cette application

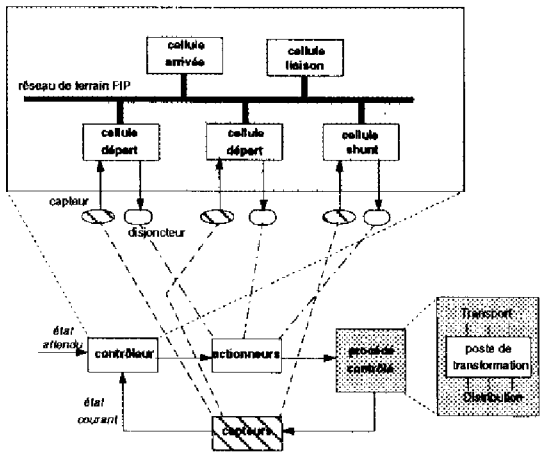

Figure 10: boucle de contrôle du système de contrôle-commande de poste réparti

met en œuvre un ensemble de cellules géographiquement éloignées de leurs équipements et dont le fonctionnement est assujetti à l'évolution dynamique des signaux électriques du procédé (intensité des lignes, tension des demi-jeux de barre). Ces cellules sont interconnectées par l'intermédiaire d'un réseau local de communication (réseau de terrain) (figure 11).

Les coopérations entre cellules, sur la base du principe de la sélectivité logique, nécessitent de mettre en œuvre des échanges de messages soumis à des contraintes temporelles fortes :

- les messages issus des cellules shunt et destinés aux cellules départ doivent permettre d'éviter ou de raccourcir la phase de confirmation au niveau des cellules départ ;
- les messages échangés entre les cellules départ doivent permettre d'éviter l'existence simultanée de plusieurs défauts départ pré-confirmés ;
- les messages envoyés par les cellules départ à une cellule arrivée doivent permettre d'éviter l'ouverture prématurée du disjoncteur arrivée.

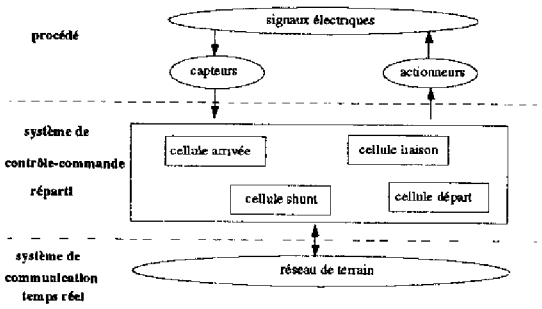

Figure 11: application de contrôle-commande temps réel distribuée

En conséquence, il est nécessaire, avant de réaliser la mise en œuvre de l'application sur le réseau de communication, de déterminer quelles sont les contraintes temporelles maximales de retard sur les échanges que l'on peut tolérer, afin de préserver le bon fonctionnement de la sélectivité logique.

2.3.2 Caractérisation des contraintes temporelles

La figure 12 présente le sous-ensemble des messages échangés entre les cellules pour les trois types de coopération (shunt–départ, départ–départ, départ–arrivée), ainsi que le sens de ces échanges (arcs en pointillé) dans le cadre de notre étude. Nous présentons maintenant les principaux scénarios illustratifs de ces coopérations et nous en déduisons les contraintes temporelles à satisfaire.

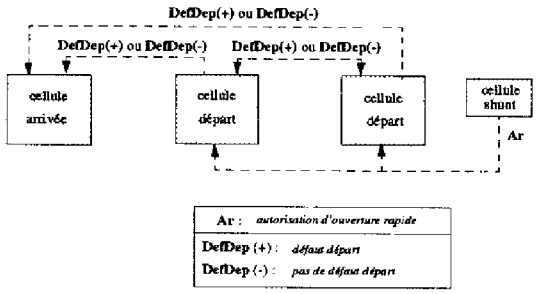

Figure 12: messages échangés entre cellules distinctes

2.3.2.1 Entre cellule shunt et cellule départ

La coopération 1 entre une cellule shunt et une cellule départ est réalisée par l'envoi, de la cellule shunt vers la cellule départ, du message **Ar**, afin d'autoriser les cellules départ en défaut à commencer leur cycle de réenclenchement rapide. Trois scénarios critiques sont présentés ci-dessous qui permettent de déterminer la borne temporelle à ne pas dépasser au plus tard pour garantir la coopération 1. Les deux premiers scénarios sont relatifs au cas des défauts monophasés et le troisième concerne le cas des défauts polyphasés.

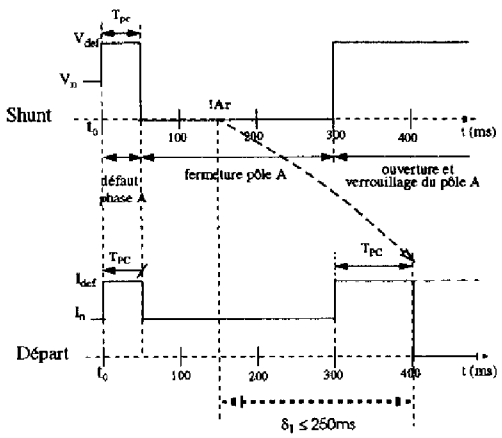

Figure 13: contrainte δ_1

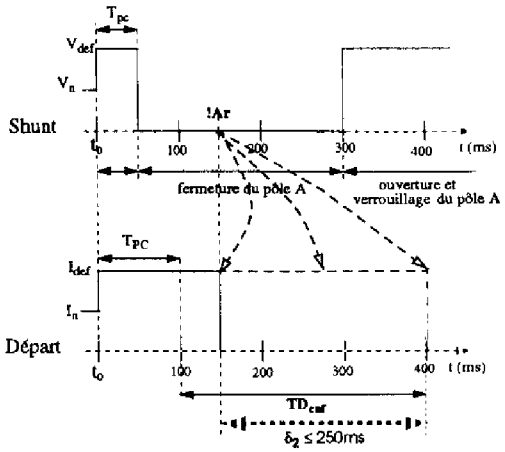

Figure 14: contrainte δ_2

Défauts monophasés

Considérons (figures 13 et 14) un défaut monophasé qui apparaît sur une phase A du shunt à l'instant t_0 et qui est conséquent de l'apparition d'un défaut départ. Ce défaut est signalé au bout du temps de pré-confirmation du shunt T_{pc} (à $t = 50\,ms$) par le capteur shunt à la cellule shunt. La cellule shunt commande tout d'abord, la fermeture de son pôle de la phase A pendant $250\,ms$ et envoie, au bout de $100\,ms$ (donc à $t = 150\,ms$ par rapport à l'instant initial du défaut), le message **Ar** à la cellule départ.

Deux cas différents doivent être considérés :

- cas 1 (figure 13) : le défaut disparaît au niveau départ pendant la fermeture du pôle de la phase A du shunt et réapparaît à l'ouverture de ce pôle (à $t = 300\,ms$) ;
- cas 2 (figure 14) : le défaut ne disparaît pas au niveau du départ (la fermeture du pôle du disjoncteur shunt n'a donc pas eu d'effet sur le défaut).

Les contraintes temporelles (appelons δ_1 et δ_2 les contraintes temporelles relatives au cas 1 et 2) sont donc les suivantes :
– dans le cas 1, on peut éviter la phase de confirmation au niveau de la cellule départ si le message **Ar** arrive avant ou au plus tard à l'instant de pré-confirmation du défaut départ (à $t = 400\,ms$), donc on a : $\delta_1 \leq 250\,ms$.
– dans le cas 2, on ne peut pas éviter la phase de confirmation ; on peut simplement la raccourcir si le message **Ar** arrive avant la fin de la phase de confirmation départ ($150\,ms \leq t \leq 400\,ms$), donc on a : $\delta_2 \leq 250\,ms$.

Défauts polyphasés

Considérons (figure 15) le cas où deux défauts distincts apparaissent simultanément sur deux phases distinctes A et B du shunt au même instant t_0. Ces défauts sont pré-confirmés au bout de T_{pc} à $t = 50\,ms$. Dans ce cas, la cellule shunt qui ne peut traiter les défauts polyphasés,

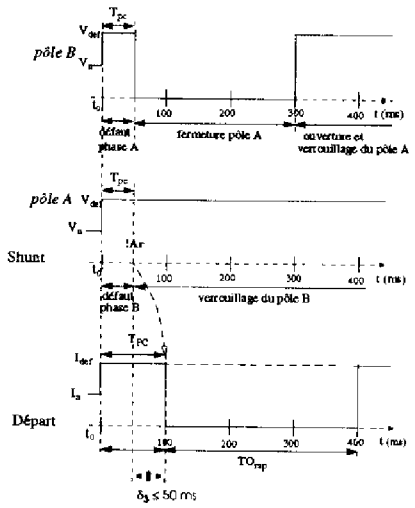

Figure 15: contrainte δ_3

envoie immédiatement (à $t = 50\,ms$) le message **Ar** aux cellules départ. Le départ a vu apparaître sur sa ligne un défaut provoqué par le cumul des défauts sur les deux phases A et B à l'instant t_0. Ce défaut est pré-confirmé au bout de T_{PC} à l'instant $t = 100\,ms$.

Dans ce cas, on peut éviter la phase de confirmation au niveau de la cellule départ si le message **Ar** arrive avant ou au plus tard à l'instant de pré-confirmation du défaut départ (à $t_0 = 100\,ms$) ; en appelant δ_3 la contrainte temporelle sur le message **Ar**, on a : $\delta_3 \leq 50\,ms$.

2.3.2.2 Entre cellule départ et cellule départ

Considérons les deux scénarios suivants :

- figure 16 : un défaut apparaît sur un départ D2 à l'instant où on a un défaut pré-confirmé sur le départ D1 (à $t = 100\,ms$) ;
- figure 17 : un défaut est pré-confirmé sur le départ D1 (à $t = 200\,ms$) au cours de son cycle de fermeture, alors qu'un défaut pré-confirmé sur le départ D2 était apparu juste à l'instant du début du cycle de fermeture (à $t = 100\,ms$).

On voit facilement que l'on peut éviter, pour chaque scénario, l'existence simultanée de deux défauts pré-confirmés (appelons δ_4 et δ_5 respectivement les contraintes temporelles associées aux messages **DefDep(+D1)** et **DefDep(+D2)** sur les figures 16 et 17) : $\delta_4 \leq 100\,ms$ et $\delta_5 \leq 100\,ms$

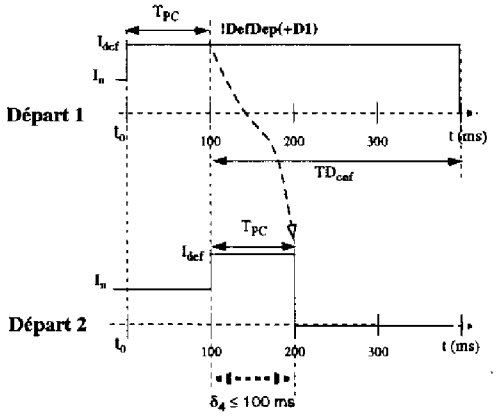

Figure 16: contrainte δ_4

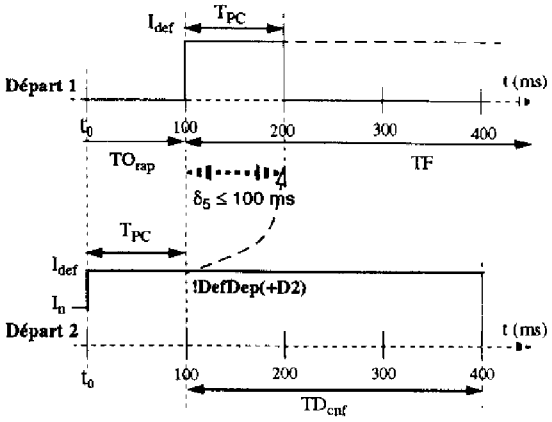

Figure 17: contrainte δ_5

2.3.2.3 Entre cellule départ et cellule arrivée

Considérons (à t_0) un défaut sur un départ D1, qui est également vu au niveau de l'arrivée (figure 18) : la cellule D1 envoie à l'instant de pré-confirmation du défaut (à $t = 100\,ms$) le message **DefDep(+D1)** à la cellule arrivée. On voit facilement que l'on peut éviter l'ouverture prématurée du disjoncteur arrivée si le message **DefDep(+D1)** est reçu par la cellule arrivée avant que sa temporisation TA_{cnf} n'arrive à son terme. En appelant δ_6 la contrainte associée au message **DefDep(+D1)**, on doit donc avoir : $\delta_6 \leq 1\,s$.

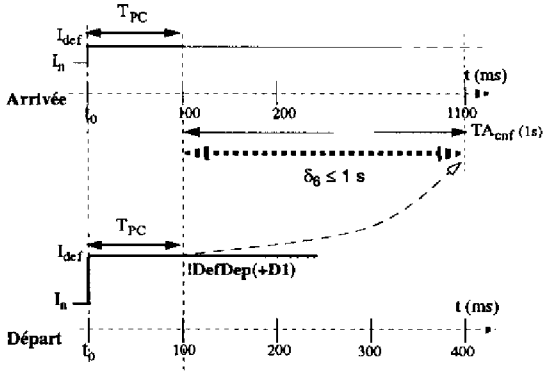

Figure 18: contrainte δ_6

2.3.2.3.1 Tableau récapitulatif

Le tableau 1 résume les contraintes temporelles identifiées précédemment et la contrainte Δ conservée pour les messages **DefDep** et **Ar**.

msg	δ
Ar	$\delta_1 = \delta_2 = 250\,ms$, $\delta_3 = 50\,ms$
DefDep	$\delta_4 = \delta_5 = 100\,ms$, $\delta_6 = 1\,s$

Tableau 1: contraintes temporelles

2.3.3 Considérations relatives au choix du réseau de communication

L'application de contrôle-commande de EDF, basée sur le principe de la sélectivité logique, est une application qui effectue la surveillance en temps réel du poste de transformation. Cette surveillance est basée sur des capteurs intelligents qui scrutent périodiquement l'état des lignes et des demi-jeux de barres car l'état du procédé doit être mesuré fréquemment pour suivre ses évolutions en temps réel. Dès qu'un état de défaillant est détecté par les capteurs intelligents, il est signalé à la cellule concernée. Il en est de même dès le rétablissement de son état normal. De plus, dans l'objectif de réaliser la surveillance globale du procédé, répartie sur l'ensemble des cellules, celles-ci doivent également s'échanger périodiquement des informations relatives à leurs états, afin de pouvoir évaluer l'état global du procédé et donc mettre en œuvre le principe de la sélectivité logique.

Le système de contrôle est donc conçu suivant le principe d'une architecture TT, guidée par le temps. En conséquence, le réseau de communication, sous-jacent au système de contrôle proprement dit, doit permettre la réalisation d'échanges périodiques. Donc un réseau de communication du type TT paraît tout indiqué. Notons encore que, outre les échanges périodiques, un réseau temps réel doit pouvoir d'une part, mettre en œuvre des mécanismes de validité temporelle sur les variables échangées et d'autre part, des échanges de messages sporadiques (alarmes...). Compte tenu de toutes ces considérations, le choix d'EDF s'est porté sur le réseau de terrain FIP.

3 Le réseau de terrain FIP

3.1 Présentation générale

Le réseau de terrain FIP [AFN90a] [Tho93] est un réseau de communication pour les niveaux bas dans la hiérarchie des systèmes de contrôle-commande (niveaux 0 et 1 de l'architecture CIM). Il permet l'interconnexion sur un bus d'équipements (appelés stations ou Processus Utilisateurs) tels que capteurs, actionneurs et automates. Il est basé sur une architecture de communication réduite aux trois couches application, liaison et physique (figure 19).

Les services de la couche application sont de deux types distincts : les services MPS (Manufacturing Periodical/Aperiodical Services) pour les échanges périodiques/apériodiques de type industriel et un sous-ensemble des services MMS (Manufacturing Message Services), essentiellement pour les échanges de gestion, de configuration et de modes de fonctionnement. Ici, nous ne considérons que les services MPS.

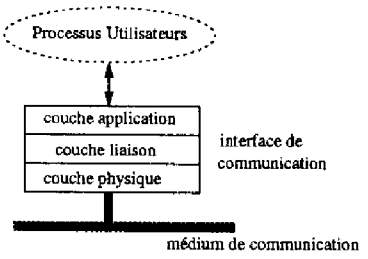

Figure 19: vue architecturale du réseau FIP

Les services MPS sont particulièrement bien adaptés pour la gestion de la Base de Données Réparties Temps Réel de FIP qui est constituée par l'ensemble des objets de couche application relatifs aux données utilisées (variable, liste de variable, variable de synchronisation). Ces services offrent aux processus utilisateurs, pour des données à contraintes temps réel, des accès périodiques et/ou apériodiques (de type Producteur/Consommateur [TN89]) dans l'interface de communication. Ils permettent, en particulier, :

- des accès en lecture et en écriture locale et/ou distante aux variables, aux listes de variables, aux synchronisations, aux indications...,
- l'élaboration de qualifieurs temporels (status de validité temporelle) en production et en consommation qui sont associés aux données de cette base.

Notons que dans le contexte MPS, les processus utilisateurs sont qualifiés avec les attributs : Producteur (producteur d'une variable), Consommateur (consommateur d'une variable) ou Tiers (ni producteur, ni consommateur, mais qui peuvent demander des mises à jour de données). Tous les objets sont identifiés par un nom unique auquel est associé un identifieur unique dans le réseau. Une variable est produite par un producteur unique et peut être utilisée par un ou plusieurs consommateurs. Chaque station possède une liste des objets qu'elle produit et qu'elle consomme.

Les services de la couche liaison de données permettent les échanges de données (valeurs des variables, status de validité associés aux variables) entre équipements producteur et consommateur, à travers un bus. L'accès au bus est régi de manière centralisé par un gestionnaire de bus, appelé Arbitre de Bus (BA)[3] qui remplit les fonctions essentielles d'arbitrage. Cette station dispose d'une liste des identifieurs des variables à faire circuler sur le médium, ainsi que les ressources nécessaires qui lui permettent d'effectuer :

- la scrutation périodique des variables périodiques (i.e. spécifiées à la configuration et déclenchées de manière autonome par le gestionnaire) par diffusion de trames requête contenant les identifieurs des variables ;
- la scrutation déclenchée apériodique de variables et de messages pour la prise en compte

[3]Toute station FIP peut effectuer les deux fonctions : Production/Consommation et Gestion d'accès. Cependant, une seule station est à un instant responsable de la gestion d'accès, ce qui permet la redondance du BA.

des requêtes de transfert venant des stations ;

- le transfert de messages avec acquittement (en point à point) ou sans acquittement (en point à point ou en diffusion).

Ici, nous ne considérons que des échanges périodiques de variables.

Le service de la couche physique assure le transfert des bits série sur paire torsadée ou fibre optique. Le support de transmission a une topologie bus et plusieurs segments de réseaux FIP peuvent être interconnectés par des répéteurs. Les vitesses de transmission existantes sont : 31,25 kb/s et 1 Mb/s (sur paire torsadée) - 2,5 Mb/s et 5 Mb/s (sur fibre optique). Le codage des données est de type MANCHESTER II [BJ88], ce qui permet de transmettre simultanément la synchronisation temporelle des signaux et les données. L'utilisation de ce codage et de délimiteurs de début et de fin de trames permettent d'assurer un certain degré de sûreté de la transmission.

3.2 La couche application

La couche application (services MPS) met en œuvre :

- des services relatifs à la gestion des objets des variables tels que : lecture et écriture locales pour les accès fréquents des processus utilisateurs ; écriture et lecture distantes pour des requêtes de mise à jour ; indication d'émission et/ou de réception ;
- des mécanismes d'élaboration de status de validité temporelle des données. Ces status sont associés aux valeurs des données dès leur production et leur consommation.

Nous allons présenter les services MPS relatifs aux échanges de variables utilisés dans le cadre de notre étude, ainsi que les status de validité temporelle.

3.2.1 Les principaux services MPS

Une Entité Application Producteur (EA-P), respectivement (noté par la suite resp.) Consommateur (EA-C), est associée à chaque Processus Utilisateur Producteur (PU-P), resp. consommateur (PU-C), d'une donnée. Nous allons présenter : les services MPS périodiques locaux qui permettent d'accéder en lecture et en écriture aux objets locaux des variables dans les interfaces de communication et les services d'indication qui permettent de savoir qu'une nouvelle valeur d'une donnée vient d'être mise à jour sur le médium.

- Le service d'écriture locale de variables permet à un PU-P d'écrire dans l'EA-P la valeur de sa variable produite. Les primitives de service offertes à l'interface PU-P/EA-P sont : *a_writeloc.rq* pour l'écriture de la variable et *a_writeloc.cnf* pour la confirmation de cette écriture.
- Le service de lecture locale de variables permet à un PU-C de lire dans l'EA-C la valeur de la variable consommée. Les primitives de service à l'interface PU-C/EA-C sont :

a_readloc.rq pour la requête de lecture et *a_readloc.cnf* pour la confirmation de cette lecture.

- Les services d'indication rendent compte aux PU-Ps et PU-Cs des transferts de variables sur le bus aux interfaces EA/PU avec :
  - le service d'indication d'émission à l'interface PU-P/EA-P. La primitive de service est *a_sent.ind* (liée au service liaison d'indication d'émission sur le médium) ;
  - le service d'indication de réception à l'interface PU-C/EA-C. La primitive de service est *a_received.ind* (liée au service liaison d'indication de réception via le médium).

Les variables échangées peuvent être des variables : de données du procédé, de synchronisation (i.e. qui ne véhiculent pas de valeur).Chaque variable est identifiée par un nom unique au niveau des PU et de la couche application et par un identifieur associé au niveau de la couche liaison.

Les variables de synchronisation ont pour rôle de synchroniser plusieurs processus répartis qui participent à une même application et dont les activités doivent être déclenchées de façon synchrone : leur exécution sur chaque site est simultanée à la réception d'une même variable de synchronisation, réalisant ainsi une application répartie synchrone. Une variable de synchronisation est produite par une entité quelquonque et ses échanges sont gérés par le BA de la même façon qune variable de données, sauf qu'elle véhicule pas de grandeur (le champ des données utile est vide). Plusieurs variables de synchronisation différentes peuvent être définies pour la synchronisation de processus utilisateurs participant à des applications réparties différentes. Ces mécanismes de synchronisation permettent d'améliorer la qualité des services rendus et, éventuellement, de réaliser une synchronisation des différentes horloges locales.

3.2.2 Les status de validité temporelle

Des mécanismes de validité temporelle sont implantés dans la couche application. Ils permettent aux Processus Utilisateurs de connaître la confiance à accorder à une variable. Est-ce qu'une variable reçue à travers le réseau a été produite à temps (*status de rafraîchissement*) et consommée à temps (*status de promptitude*) ? Par le biais de ces qualifieurs temporels (ou status de validité temporelle), un équipement consommateur peut déterminer si une variable n'est pas *périmée* parce qu'elle a été produite trop tard et/ou parce qu'elle a été consommée trop tard par rapport à des évènements de référence (tels que la dernière production, consommation, ou réception de variable de synchronisation...).

- Le rafraîchissement est une valeur booléenne (Vrai ou Faux) relative à la validité de production par le PU-P dans une fenêtre temporelle donnée. Il est associé à la variable produite au niveau de l'EA-P ;

- La promptitude est une valeur booléenne (Vrai ou Faux) relative à la validité de consommation par le PU-C dans une fenêtre temporelle donnée. Elle est associée à la variable consommée dans l'EA-C lors de sa lecture par le PU-C.

Les fenêtres temporelles sont spécifiées à la configuration et leur durée dépend des caractéristiques de production et de consommation des PUs répartis. De plus, ces qualifieurs temporels peuvent, en particulier, posséder le caractère :

- asynchrone lorsque la fenêtre temporelle et le status évoluent sur la base des seuls évènements liés à la production, resp. réception, de la variable. Le status vrai signifie que la production, resp. consommation, a eu lieu depuis moins de T unités de temps (T étant la durée de la fenêtre temporelle, i.e. la durée de validité associée à la valeur de la variable par le concepteur). Dans ce cas, le status asynchrone indique, non pas un mode de fonctionnement périodique, mais le temps maximum séparant deux occurrences normales, soit d'écriture, soit de lecture, d'une nouvelle valeur de la variable ;

- synchrone lorsqu'ils sont élaborés à partir des réceptions de variables de synchronisation. La fenêtre temporelle et le status évoluent sur la base d'indication de réception de variable de synchronisation. Le status vrai signifie que la production, resp. consommation, date au plus tard de T unités de temps par rapport au dernier évènement de synchronisation.

D'autres status (asynchrones, ponctuels, resynchronisés) sont définis dans la norme [AFN90b]. Citons également les travaux de Lorenz [Lor94] [LMT94] qui a défini de nouveaux status de validité temporelle pour le réseau de terrain FIP.

3.3 La couche liaison de données

La couche liaison de données est constituée d'un ensemble de buffers de données produites et/ou consommées qui contiennent les dernières valeurs mises à jour, soit par l'utilisateur (dans le cas de données produites), soit par le réseau (dans le cas de données consommées). L'écriture d'une nouvelle valeur dans un buffer écrase l'ancienne. L'accès à un buffer s'effectue par l'intermédiaire de l'identifieur qui lui est associé.

3.3.1 Services de la couche liaison

La couche liaison met à disposition de la couche application l'ensemble des services suivants : lecture et écriture locales et/ou distantes de buffer, transfert de buffer, demandes explicites libres ou spécifiées de transfert de buffer, etc.

Les services d'écriture et de lecture locales sont locaux à une entité liaison et ils ne génèrent pas d'activités sur le médium de communication. Nous présentons ici les trois services qui sont considérés dans la suite de notre étude.

- Le service d'écriture locale de buffer permet à une Entité Application Producteur (EA-

P) d'écrire, dans le buffer de l'Entité Liaison Producteur (EL-P), les données reçues de la couche application relatives à la variable produite. Les primitives sont : *l_put.rq* pour la requête d'écriture et *l_put.cnf* pour la confirmation d'écriture.

- Le service de lecture locale de buffer permet à une Entité Application Consommateur (EA-C) de lire les données contenues dans le buffer de l'Entité Liaison Consommateur (EL-C) et reçues via le réseau. Les primitives sont : *l_get.rq* pour la requête de lecture et *l_get.cnf* pour la confirmation de lecture.
- Le service de transfert de buffer permet à une EL-P (resp. EL-C) de signaler à une entité EA-P (resp. EA-C) l'envoi (resp. la réception) d'une trame contenant la variable produite (resp. consommée). Les primitives sont : *l_sent.ind*, respectivement *l_received.ind.*

Les services locaux d'écriture et de lecture de buffer sont totalement asynchrones par rapport au fonctionnement du protocole de la couche liaison. Par contre, le service de transfert de buffer est lié au protocole.

3.3.2 Le protocole de la couche liaison

Le contrôle de l'accès au médium de FIP est basé sur une technique classique de contrôle centralisé. C'est l'Arbitre de Bus (BA) qui donne le droit de parole aux différentes stations par scrutation. FIP est donc basé sur un protocole à accès centralisé à échanges prédéterminés. Ceci implique que tous les échanges soient pré-définis et placés dans une table de scrutation qui comprend l'ensemble des requêtes d'échanges périodiques à effectuer. Ceci n'est possible que pour des applications relativement figées (qui n'évoluent pas dans le temps[4]).

3.3.2.1 Les échanges de protocole

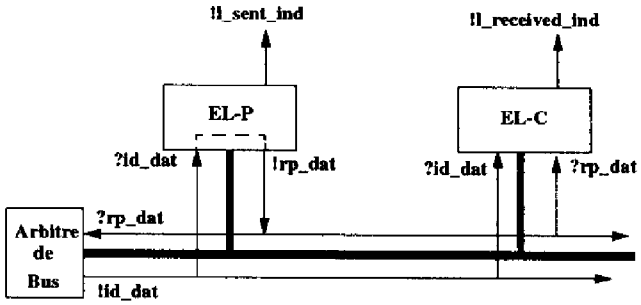

Figure 20: échanges sur le médium

Les échanges de protocole présentés ici concernent les services périodiques de transfert de buffers pour la mise à jour, en temps réel, de la base de données répartie. Ces échanges se

[4] Il est tout de même possible de changer de mode opératoire en changeant de table de scrutation.

déroulent selon les étapes suivantes (figure 20) :

A) la diffusion périodique sur le bus, par le BA, d'une trame requête spécifiant l'identifieur de la donnée à échanger **!id_dat**. L'EL-P et les EL-Cs concernés par cette donnée sont sensibilisées lorsqu'elles reçoivent cette trame ;

B) l'EL-P, au bout de son Temps de Retournement, diffuse sur le bus la trame réponse avec le contenu du buffer de couche liaison de l'identifieur demandé **!rp_dat**. Le service d'indication d'émission à l'interface EL-P/EA-P signale alors cette émission sur le médium à la couche application par l'envoi de la primitive **!l_sent.ind** ;

C) lorsque les EL-Cs reçoivent la trame réponse **?rp_dat**, ils mémorisent les données reçues dans leur buffer de couche liaison. Le service d'indication de réception signale cette réception à la couche application (interface EL-C/EA-C) par l'envoi de la primitive **!l_received.ind**. Le BA reçoit également la trame réponse, ce qui lui permet de vérifier le bon déroulement de l'échange en cours et de poursuivre les émissions de requêtes d'échanges. Les échanges périodiques continuent alors au point A avec l'émission d'une nouvelle donnée (la suivante dans la table de scrutation).

Remarque : Notion de transaction élémentaire
Une transaction élémentaire est définie, dans le réseau FIP, comme l'ensemble des échanges nécessaires à la mise à jour d'une donnée v périodique. Elle comprend : la durée d'émission de la trame requête *id_dat* ; le temps de retournement de la station productrice de la donnée demandée v ; la durée d'émission de la trame réponse *rp_dat* ; le temps de retournement du BA avant l'émission de la trame requête de l'identifieur suivant.
La durée d'une transaction élémentaire dépend, en plus, : du nombre d'octets, noté *oct* (soit $8 * oct$ bits), utilisé pour les données utiles de niveau utilisateur (sachant qu'au total 61 bits sont rajoutés par l'ensemble des couches) ; du temps $TMAC$ (1Mb/s) nécessaire pour la transmission d'un bit d'information sur le médium et du temps de retournement TR d'une station ($10 * TMAC \leq TR \leq 70 * TMAC$). Le temps d'une transaction élémentaire TE_{oct} est donc : $TE_{oct} = (2TR + 8 * oct + 122)TMAC$.

3.3.2.2 La table de scrutation de l'arbitre de bus

La table de scrutation du BA est définie hors ligne comme un échéancier des échanges à effectuer. Elle contient la liste de tous les identifieurs des données périodiques et apériodiques connues à échanger pour la réalisation des communications temps réel.

Cet échéancier est défini comme un Macro-cycle (MC) composé d'une séquence de micro-cycles (μCs) (ou cycles élémentaires). Le MC est répété indéfiniment par le BA et un changement de MC peut être effectué en ligne (s'il y a modification des besoins périodiques de l'application). Les séquences d'échanges périodiques sont généralement définies de façon statique et les échanges apériodiques de façon dynamique (par les requêtes des processus application).

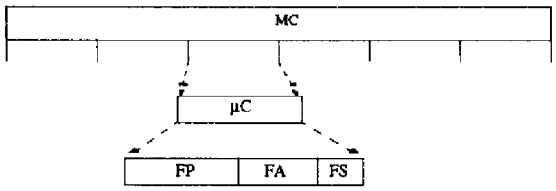

Figure 21: description du Macro-Cycle

Chaque micro-cycle est composé (figure 21) : d'une fenêtre périodique (FP) destinée aux échanges périodiques ; d'une fenêtre apériodique (FA) destinée aux échanges apériodiques ; et d'une fenêtre de synchronisation ou bourrage (FS) destinée aux échanges de bourrage[5].

Les durées du macro-cycle et des micro-cycles sont définies à partir des périodes (p_i) de l'ensemble des variables périodiques (v_i) telles que : $P_{\mu C} = pgcd(p_i)$ et $P_{MC} = ppcm(p_i)$.

Ainsi, le modèle Producteur/Distributeur/Consommateur de communication de FIP est bien adapté aux trafics horizontaux temps réel : par la mise à jour périodique de la base de données répartie (constituée par l'ensemble des données localisées dans les interfaces de communication producteur et consommateur) ; par les mécanismes temporels associés aux variables qui sont offerts. Les échanges en diffusion, s'effectuant généralement sans acquittement, permettent d'assimiler le réseau FIP à une base de données mise à jour en temps réel avec la possibilité d'utiliser les status de validité temporelle comme mécanismes de détection d'erreur de production et/ou de transmission (basés sur la gestion de la durée de vie limitée). Ainsi, les services périodiques de mise à jour locaux et de transfert de buffer par diffusion de FIP sont spécifiquement bien adaptés pour la réalisation des échanges à contraintes temporelles strictes.

3.4 Sur la mise en œuvre de l'application de contrôle-commande de EDF sur le réseau FIP

3.4.1 Etapes de la mise en œuvre

On peut identifier deux grandes étapes :

1. la première étape concerne la détermination des périodes de mise à jour des variables sur le réseau ;
2. la deuxième étape concerne la détermination de la table de scrutation de l'Arbitre de Bus.

3.4.2 Considérations relatives au choix des périodes de mise à jour sur le réseau

[5] pour garantir des micro-cycles de durées égales.

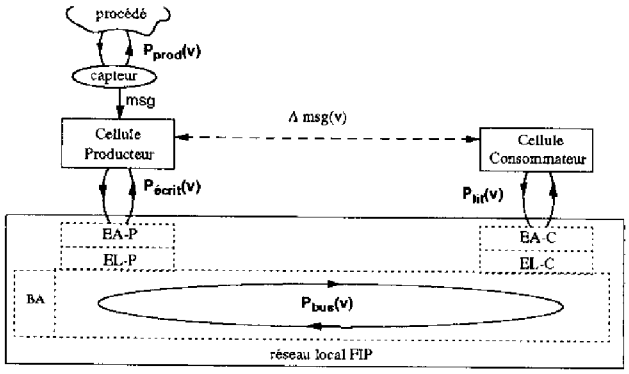

Figure 22: différentes fréquences associées à une donnée *msg*

Le schéma représenté sur la figure 22 permet de situer la problématique (nous n'avons pas représenté l'actionneur situé côté consommateur car il n'intervient pas dans la réflexion présentée).

Notons que nous ne considérons pas, tout d'abord, les temps de propagation introduits par le réseau de terrain FIP. Cette hypothèse est raisonnable car le temps de propagation dans les réseaux locaux (de l'ordre de $0.05\,\mu s$ pour 1 km) est très très faible devant les grandeurs des contraintes temporelles de l'application (la plus petite est $50\,ms$).

La mise à jour d'une variable v sur le réseau est caractérisée par trois composantes qui ont toutes la même valeur (en considérant le réseau FIP avec les status de validité) :

- la période d'écriture de la variable v dans l'EA-P par la cellule producteur ; appelons $\mathbf{P}_{écrit}(\mathbf{v})$ cette période ;
- la période des échanges sur le bus du message transportant cette variable v ; appelons $msg(v)$ et $\mathbf{P}_{bus}(\mathbf{v})$ respectivement ce message et la période d'échange de ce message sur le bus ;
- la période de lecture de la variable v dans l'entité EA-C par la cellule consommatrice ; appelons $\mathbf{P}_{lit}(\mathbf{v})$ cette période.

Le choix de ces périodes dépend de deux contraintes :

- la période de production de la variable v (notée $\mathbf{P}_{prod}(\mathbf{v})$), qui correspond à la période avec laquelle le capteur mesure la valeur de la variable v indiquant l'état courant du procédé. Cette période est fixée par la dynamique du procédé car le capteur doit mesurer son état aussi rapidement que ses évolutions, c'est-à-dire *en temps réel*. Ensuite, le capteur intelligent informe la cellule producteur d'un changement d'état (défaillant, non défaillant) par un message noté **msg**, et transmis de façon évènementielle) ;
- la contrainte temporelle associée au message $msg(v)$ qui doit véhiculer la variable v depuis la cellule Producteur vers la (les) cellule(s) Consommateur ; appelons $\Delta\ msg(v)$ cette contrainte temporelle qui résulte du principe de la sélectivité logique et qui a été

évaluée dans le cadre de l'analyse présentée au paragraphe 2.3.

Deux cas doivent être analysés :

- soit $\mathbf{P_{prod}(v)} > \Delta\ msg(v)$: dans ce cas, les périodes de mise à jour sur le réseau de la variable v résultent de la considération de $\Delta\ msg(v)$. Plus précisément, il faut définir la période $\mathbf{P_{bus}(v)}$ en considérant que, pour augmenter la "disponibilité temps réel" des données, la période de diffusion sur le bus doit respecter le théorème de Shannon (défini dans la théorie des systèmes échantillonnés [Y. 69]) qui indique que : *la période du signal transmis doit être au moins deux fois inférieure à la période du signal observé.* On choisit donc :

$$P_{bus}(v) \leq \frac{\Delta\ msg(v)}{r} \quad avec \quad r > 2 \tag{3.1}$$

  La valeur du facteur r est laissée au choix du concepteur lors de son étude. Soulignons cependant, que plus r est grand, plus le retard entre la production et la mise à disposition de la dernière valeur pour l'utilisateur consommateur est réduit.

  Le tableau 2 résume les périodes conseillées (établies avec $r = 2$) pour la mise à jour sur le médium des variables périodiques considérées dans cette étude.

Messages	Périodes sur le médium (ms)
Ar	$\mathbf{P_{bus}(Ar) \leq 25}$
DefDep	$\mathbf{P_{bus}(DefDep) \leq 50}$

Tableau 2: périodes de mise à jour sur le réseau

- soit $\mathbf{P_{prod}(v)} < \Delta\ msg(v)$: dans ce cas, on choisit :

$$P_{bus}(v) = P_{écrit}(v) = P_{lit}(v) \leq P_{prod}(v) \tag{3.2}$$

  Les contraintes temporelles $\Delta\ msg(v)$ sont alors forcément satisfaites.

Remarque : en toute exactitude, il faudrait également tenir compte des temps introduits par les entités de communication et les durées de transfert sur le réseau.

3.4.3 Ordonnançabilité

L'étude de l'ordonnancement (ordonnançabilité) est une étape importante dans la phase de mise en œuvre d'une application temps réel répartie car d'elle dépend la capacité du réseau à satisfaire les besoins de communication temps réel. Elle a pour objectif de déterminer la table de scrutation du BA.

Afin de garantir la capacité du réseau FIP à assurer la mise en œuvre d'une application temps réel répartie, l'ordonnancement des échanges périodiques sur les micro-cycles doit être construit en respectant les contraintes temporelles fortes des échanges périodiques sans surcharger les fenêtres périodiques des micro-cycles (μCs) et ainsi, laisser du temps libre pour les échanges apériodiques qui seraient nécessaires.

L'étude de l'ordonnancement doit également considérer les critères essentiels qui sont :

- le codage des données. Il est nécessaire pour cela de préciser, avant d'effectuer l'ordonnancement, quelle est la nature des données (booléen, integer, tabular...) et de dire sur combien d'octets (*oct*) l'information utile de niveau utilisateur est codée. En effet, la durée d'une transaction élémentaire TE_{oct} dépend : de ce codage, du temps de retournement des stations et de la vitesse de transmission sur le médium.

- le taux d'utilisation du médium. De par la décomposition de la table de scrutation en fenêtres temporelles (micro-cycles) de durées fixes, nous considérons qu'un *bon ordonnancement* ne doit pas surcharger chaque micro-cycle (et donc le macro-cycle) au-delà d'une certaine limite (que nous fixons autour de 60% à 70% de la durée d'un micro-cycle) pour les échanges périodiques. Cette borne a pour but de laisser suffisamment de temps pour pourvoir satisfaire les requêtes d'échanges apériodiques dynamiques (urgentes ou non) au cours des différents micro-cycles, sans retard. Cependant, cette limitation n'est pas toujours réalisable si les besoins en échanges périodiques sont nombreux et de fréquences élevées. Alors, si c'est possible, il faut déterminer un nouveau codage et/ou un nouvel ordonnancement en changeant de politique, dans l'objectif de diminuer la charge des micro-cycles. Le calcul de la charge de chaque micro–cycle est obtenu en faisant la somme des temps de transaction élémentaire des données périodiques v_i de chaque micro–cycle μC_k (où k est l'ordre du micro–cycle dans le macro-cycle et tel que $(k \in \{1, ..., K = \frac{ppcm}{pgcd}(p_i)\})$: $\sigma(\mu C_k) = \sum_{v_i \in \mu C_k} TE_{oct}(v_i)$. Le taux d'utilisation de chaque micro-cycle d'ordre k, noté $\tau(\mu C_k)$, est alors : $\tau(\mu C_k) = \frac{\sigma(\mu C_k)}{P_{\mu C}}$.

L'état actuel de la norme de FIP ne prévoit pas d'algorithmes d'ordonnancement. Cependant, des travaux ont déjà été effectués pour adapter les algorithmes classiques d'ordonnancement tels que Rate Monotonic (RM) ou Earliest Deadline (ED) [LL73] aux cas des réseaux tels que FIP [RN93]. Nous avons également proposé un algorithme d'ordonnancement[6] basé sur des techniques de répartition de charges qui permet une assignation équilibrée des variables parmi l'ensemble des micro–cycles. Cet algorithme est présenté dans [BJS95a]. L'utilisation de cet algorithme a permis de déterminer un ordonnancement satisfaisant les contraintes temporelles fortes en considérant l'ensemble des échanges périodiques nécessaires (de l'ordre de 115 variables périodiques) pour une topologie classique de poste de transformation d'énergie (comprenant environ 31 cellules distinctes). De plus, il a été montré que le taux d'utilisation du réseau dépend fortement du choix du codage et du temps de retournement TR des stations (pour une vitesse de transmission classique de 1Mb/s et pour des périodes de mise à jour réseau fixée avec $\tau = 4$). Un "bon" compromis entre le codage et les périodes de mise à jour est proposé afin d'éviter des situations de surcharge des micro-cycles.

[6] Ce travail ne faisant pas partie intégrante de l'objectif de ce manuscrit, mais se situant en marge, nous ne le présentons pas ici.

4 Conclusion

L'objectif de ce chapitre était double :

- présenter le fonctionnement du poste de transformation d'énergie électrique HTB/HTA (de manière plus digeste et plus structurée pour les "non initiés" par rapport aux documents EDF dont nous disposions) et plus particulièrement, les mécanismes de la sélectivité logique (pour la localisation et le traitement des défauts) en terme d'application distribuée temps réel ;
- décrire les principales caractéristiques (services et mécanismes) du réseau FIP permettant la mise en oeuvre de l'application temps réel de contrôle-commande du poste HTB/HTA et également, préciser la démarche de définition des fonctionnalités de base de l'Arbitre de Bus (périodicité et ordonnancement des différents échanges sur le bus), compte tenu des caractéristiques temporelles et de configuration de l'application.

En ce qui concerne la sélectivité logique, nous voulons insister sur : d'une part, la présentation des différentes coopérations (coopérations shunt-départ, départ-départ, départ-arrivée) qui a pour objectif de poser la problématique de la répartition des automatismes de commande et de surveillance en vue d'identifier les besoins en communication ; d'autre part, la détermination des contraintes temporelles des communications entres les sites distants qu'il est absolument nécessaire de respecter, afin de garantir les fonctionnalités essentielles de l'application. En particulier, nous avons déterminé les contraintes temporelles sur les transferts des messages échangés pour la réalisation des coopérations entre cellules shunt-départ (afin que les cellules départ raccourcissent la phase de confirmation) et départ–départ (afin d'avoir la propriété d'évitement de défaut multiples).

En ce qui concerne le réseau de terrain FIP, nous avons présenté d'une part, les principaux services que nous estimons nécessaire d'utiliser pour la mise en œuvre du principe de la sélectivité logique dans le cadre de la surveillance du poste HTB/HTA (lecture et écriture locales, indication d'émission et de réception) et d'autre part, les mécanismes de validité temporelle (le rafraîchissement et la promptitude) associés à ces services qui sont des spécificités du réseau FIP et qui sont des mécanismes essentiels pour caractériser la qualité de service (l'attribut synchrone résulte du choix que nous avons fait pour une architecture guidée par le temps). Nous avons également défini une méthode de calcul de la période nécessaire aux échanges sur le bus du réseau FIP afin que les contraintes temporelles imposées par les coopérations inter-cellules soient satisfaites (utilisation du théorème de Shannon pour la reconstitution de l'état d'une cellule dans une autre cellule).

Chapitre 4

Le réseau de terrain FIP : structuration, modélisation et analyse

1 Introduction

Ce chapitre présente l'application de la méthodologie, décrite au chapitre 2, au réseau de terrain FIP et plus précisément à la couche application. Nous considérons deux Processus Utilisateurs : un Processus Utilisateur Producteur (PU-P) et un Processus Utilisateur Consommateur (PU-C), qui participent à une même activité répartie synchrone et qui, pour ce faire, utilisent les services de deux entités de la couche application de FIP. Ces deux Entités Application, respectivement Producteur et Consommateur, mettent en œuvre (cf. chapitre 3) les services périodiques d'écriture locale de variable, de lecture locale de variable et d'indication, en implantant les mécanismes de validité temporelle (rafraîchissement, promptitude) synchrones. La synchronisation des processus utilisateurs est effectuée par une variable de synchronisation dont la période de diffusion est égale à la période de la variable de données qui transporte les informations utilisateurs. La couche liaison de FIP met en œuvre (cf. chapitre 3) les services d'écriture locale de buffer, de lecture locale de buffer et de transfert de buffer. La couche physique n'est pas considérée dans notre étude.

Ce chapitre comprend trois parties :

- la première partie présente la structuration du système étudié ;
- la deuxième partie présente la modélisation du système sur la base du modèle Réseaux de Petri Temporisés Stochastiques ;
- la troisième partie présente l'analyse du comportement de la couche application de FIP et plus précisément, l'analyse de la qualité des services associés aux mécanismes de rafraîchissement et de promptitude (analyses qualitatives et quantitatives).

2 Structuration

L'architecture globale est celle déjà présentée sur la figure 4 du chapitre 2. Nous développons ici l'architecture détaillée des couches application et liaison du réseau FIP (modules et échanges).

2.1 Couche application

Nous avons donc deux Entités Application : l'Entité Application Producteur (EA-P), associée au PU-P du site producteur ; l'Entité Application Consommateur (EA-C), associée au PU-C du site consommateur. Les architectures détaillées de ces Entités sont maintenant présentées en faisant référence à l'architecture générale d'une Entité Application de site utilisateur définie au chapitre 2 (cf. figure 5).

2.1.1 Entité Application Producteur (EA-P)

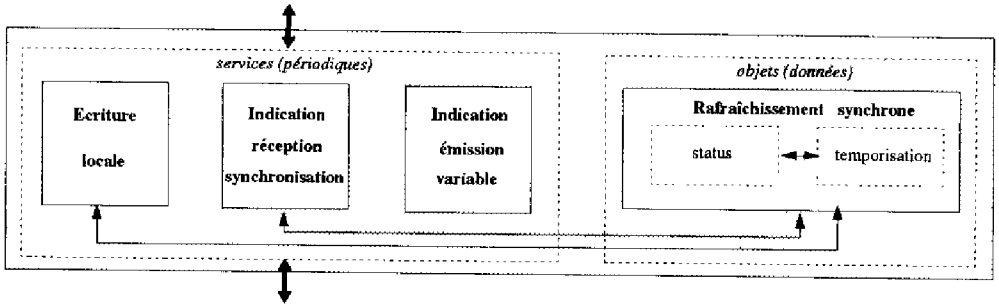

Figure 1: architecture détaillée de l'EA-P

La figure 1 représente l'architecture détaillée de l'EA-P qui comprend deux modules de plus haut niveau : le module *services (périodiques)* et le module *objets (données)*.

Le module *services (périodiques)* comprend les trois modules élémentaires : **Écriture locale**, **Indication réception synchronisation** et **Indication émission variable**.

Le module *objets (données)* comprend le module élémentaire : **Rafraîchissement synchrone**, composé de deux parties, Status et Temporisation.

Les flèches à l'intérieur de l'EA-P indiquent les modules élémentaires en relation. Les modules *Écriture locale* et *Rafraîchissement synchrone* sont en relation pour associer la valeur du status de rafraîchissement à la valeur de la variable produite lors d'une écriture. Les modules *Indication réception synchronisation* et *Rafraîchissement synchrone* sont en relation pour réaliser les actions de mise à jour du status de rafraîchissement et de la temporisation associée, lors de la réception d'une variable de synchronisation.

Les mécanismes d'évolution du status de rafraîchissement sont représentés sur la figure 2.a et sont basés sur les points suivants [AFN90b] : 1) réception de la variable de synchronisation,

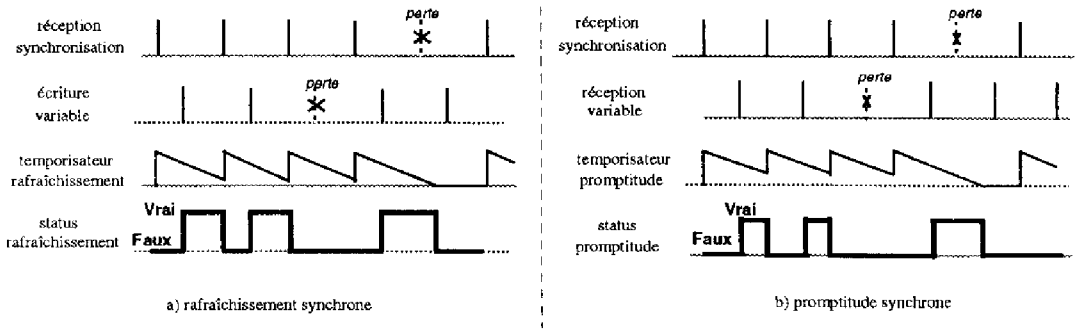

Figure 2: status de validité temporels synchrones

ce qui provoque l'initialisation ou la réinitialisation du temporisateur de rafraîchissement ; 2) écriture de la variable (portant les données utilisateurs) dans l'EA-P, ce qui provoque la mise à Vrai du status de rafraîchissement s'il est Faux et si la temporisation est On (pas de modification du status si la temporisation est Off) ; 3) mise à Faux du status à la fin du délai de la temporisation, ceci résultant d'une non-réception (perte) de la variable de synchronisation.

Le status de rafraîchissement synchrone Vrai signifie que la variable a bien été produite dans la fenêtre temporelle relative à l'évènement correspondant de synchronisation.

2.1.2 Entité Application Consommateur (EA-C)

La figure 3 représente l'architecture détaillée de l'EA-C qui comprend deux modules de plus haut niveau : le module *services (périodiques)* et le module *objets (données).*

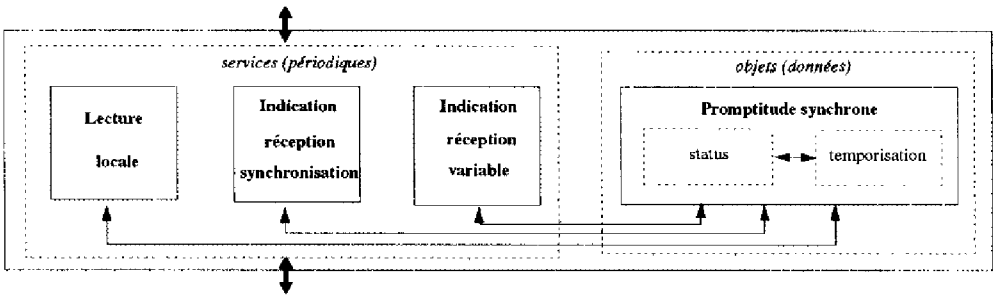

Figure 3: architecture détaillée de l'EA-C

Le module *services (périodiques)* comprend les trois modules élémentaires : **Lecture locale**, **Indication réception synchronisation** et **Indication réception variable.**

Le module *objets (données)* comprend le module élémentaire : **Promptitude synchrone**, composé de deux parties, Status et Temporisation.

Les flèches à l'intérieur de l'EA-C indiquent les modules élémentaires en relation. Les modules *Lecture locale* et *Promptitude synchrone* sont en relation pour associer la valeur du status de promptitude à la valeur de la variable reçue lors d'une demande de lecture locale par l'EA-C. Les modules *Indication réception synchronisation* et *Promptitude synchrone* sont en relation pour réaliser les actions de mise à jour du status de promptitude et de la temporisation associée, lors de la réception d'une variable de synchronisation.

Les mécanismes d'évolution du status de promptitude sont représentés sur la figure 2.b et sont basés sur les points suivants : 1) réception de la variable de synchronisation, ce qui provoque l'initialisation ou la réinitialisation du temporisateur de promptitude ; 2) réception de la variable via le médium, ce qui provoque la mise à Vrai du status de promptitude s'il est Faux et si la temporisation est On (pas de modification du status si la temporisation est Off) ; 3) mise à Faux du status à la fin du délai de la temporisation, ceci résultant d'une non-réception (perte) de la variable de synchronisation.

Le status de promptitude Vrai signifie que la variable a bien été reçue dans la fenêtre temporelle relative à l'évènement correspondant de synchronisation.

2.2 Couche liaison

Nous considérons ici seulement une abstraction de la couche liaison, c'est-à-dire les modules *contrôle logique* de ces entités. En effet, l'accès au médium a déjà été étudié dans des travaux antérieurs [Ata94] [JABF92]. Nous présentons donc l'architecture détaillée des modules *contrôle logique* des Entités Liaison Producteur (EL-P) et Consommateur (EL-C), ainsi que de l'Entité Liaison de l'Arbitre de Bus (EL-BA).

Les architectures détaillées de ces Entités sont présentées en faisant référence aux architectures générales d'une Entité Liaison de site utilisateur et du site gestionnaire, données au chapitre 2 respectivement sur les figures 6 et 7.

2.2.1 Entité Liaison Producteur (EL-P)

La figure 4 représente l'architecture détaillée du bloc *contrôle logique* de l'EL-P qui comprend deux modules de plus haut niveau : *services (périodiques)* et *mémorisation.*

Le module *services (périodiques)* comprend les trois modules élémentaires : **Écriture locale buffer**, **Indication réception synchronisation** et **Indication émission variable.**

Le module *mémorisation* comprend le module élémentaire : **Buffer producteur.**

Les flèches à l'intérieur de l'EL-P indiquent les modules élémentaires en relation. Les modules élémentaires *Écriture locale buffer* et *Buffer producteur* sont en relation pour réaliser l'écriture des données (valeur de la variable et status de rafraîchissement) dans le buffer producteur par le service d'écriture locale. Les modules *Indication émission variable* et *Buffer producteur* sont en relation pour réaliser le service de transfert périodique du buffer.

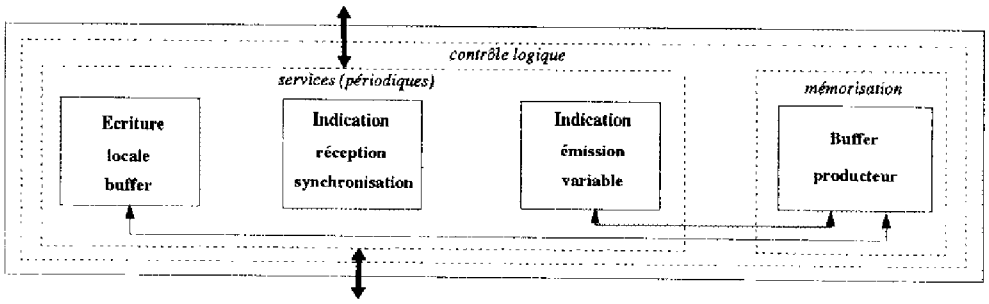

Figure 4: architecture détaillée de l'EL-P

2.2.2 Entité Liaison Consommateur (EL-C)

La figure 5 représente l'architecture détaillée du bloc *contrôle logique* de l'EL-C qui comprend deux modules de plus haut niveau : *services (périodiques)* et *mémorisation.*

Le module *services (périodiques)* comprend les trois modules élémentaires : **Lecture locale buffer, Indication réception synchronisation** et **Indication réception variable.**

Le module *mémorisation* comprend le module élémentaire : **Buffer consommateur.**

Les flèches à l'intérieur de l'EL-C indiquent les modules élémentaires en relation. Les modules élémentaires *Lecture locale buffer* et *Buffer consommateur* sont en relation, par le service de lecture locale, pour accéder aux données reçues (valeur de la variable et status de rafraîchissement) contenues dans le buffer consommateur. Les modules *Indication réception variable* et *Buffer consommateur* sont en relation pour réaliser le service de transfert périodique du buffer.

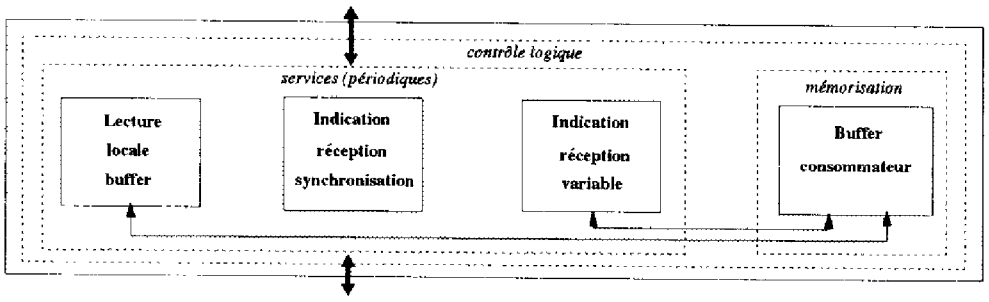

Figure 5: architecture détaillée de l'EL-C

2.2.3 Entité Liaison de l'Arbitre de Bus (EL-BA)

La figure 6 représente l'architecture détaillée du bloc *contrôle logique*, restreint au bloc *services de gestion*, de l'EL-BA, comprenant uniquement le module : *génération flux périodique.*

Ce module **génération flux périodique** est restreint à l'automate d'envoi des trames de requêtes sur le bus (qui résulte de l'ordonnancement, défini hors ligne, des échanges pério-

diques).

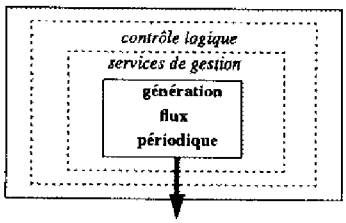

Figure 6: architecture détaillée de l'EL-BA

2.3 Échanges

Nous résumons cette présentation en décrivant :

- tout d'abord, les échanges de services puis, les échanges internes, relativement à l'EA-P et à l'EA-C (on pourrait faire de même pour les autres entités) ;
- ensuite, la vue globale des échanges de services et protocoles dans l'architecture de communication entre les sites : gestionnaire, producteur et consommateur.

2.3.1 Échanges relatifs aux entités Application EA-P et EA-C

La variable de donnée est identifiée aux niveaux des processus utilisateurs et de la couche application, par son nom unique *var* et au niveau de la couche liaison par l'identificateur global *id_v* qui lui est associé. De même, la variable de synchronisation est identifiée par son nom global *syn* au niveau couche application et processus utilisateurs et par son identificateur *id_s* au niveau couche liaison.

2.3.1.1 Entité Application EA-P

Les échanges relatifs à une EA-P sont explicités et représentés sur la figure 7 : d'une part, aux interfaces PU-P/EA-P et EA-P/EL-P en ce qui concerne les services et d'autre part, en interne entre les modules élémentaires de l'EA-P.

En ce qui concerne les services :

- le module **Écriture locale** gère :
  – à l'interface avec le PU-P, l'écriture de la valeur *val* d'une variable *var* venant du PU-P (`a_writeloc.rq(var,val)`) et la confirmation de cette lecture (`a_writeloc.cnf`) ;
  – à l'interface avec l'EL-P, l'écriture dans le buffer liaison (après l'association de la valeur du status de rafraîchissement *raf* à la valeur de la variable) du couple $[val, raf]$, appelé [data], et identifié par son identificateur *id_v* (`l_put.rq(id_v,[data])`), ainsi que la confirmation liaison de cette écriture (`l_put.cnf(id_v)`) ;

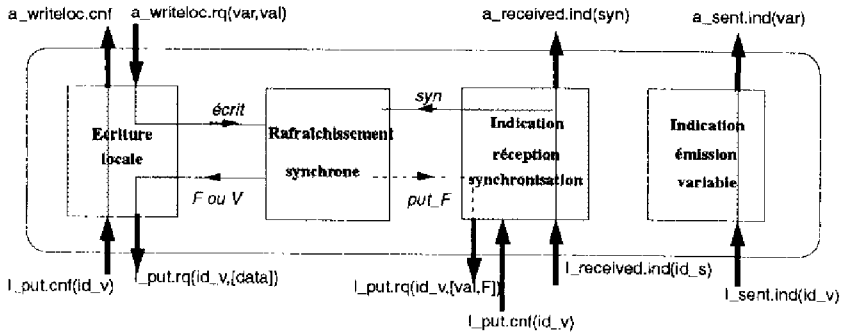

Figure 7: vue des échanges au niveau de l'EA-P

- le module **Indication réception synchronisation** gère :
  - à l'interface avec l'EL-P : la réception de la variable de synchronisation, identifiée par son identificateur *id_s* (`l_received.ind(id_s)`) ; l'écriture locale du buffer liaison, suite à une mise à Faux du status (`l_put.rq(id_v,[val,F])`) et sa confirmation (`l_put.cnf(id_v)`), provoquée soit, par la réception de la synchronisation soit, par la fin de la temporisation, due à la non réception de la synchronisation ;
  - à l'interface avec le PU-P, l'envoi de l'indication de réception de la variable de synchronisation *syn* au PU-P (`a_received.ind(syn)`) ;

- le module **Indication émission variable**, qui a un rôle élémentaire de signalisation pour le PU-P, gère :
  - à l'interface EL-P, la réception de l'indication d'émission de la variable (`l_sent.ind(id_v)`) ;
  - à l'interface EA-P, l'envoi de l'indication d'émission de la variable (`a_sent.ind(var)`) ;

- le module **Rafraîchissement synchrone** n'est pas concerné par les interactions de services avec les modules des niveaux adjacents.

En ce qui concerne les échanges internes, nous détaillons les quatre types d'échanges (*écrit*, *F* ou *V*, *syn*, *put_F*) qui sont introduits :

- l'échange *écrit* entre le module **Écriture locale** et le module **Rafraîchissement synchrone** est consécutif à la réception de la primitive d'écriture locale par le PU-P et représente la mise à jour du status. Le résultat de l'état courant du status est alors transmis du module **Rafraîchissement synchrone** vers le module **Écriture locale** pour indiquer à ce dernier si le status est Vrai ou Faux par les messages respectivement, *V* ou *F* ;

- l'échange *syn* entre le module **Indication réception synchronisation** et le module **Rafraîchissement synchrone** est consécutif à la réception de la variable de synchronisation par le module **Indication réception synchronisation** ; cet échange peut provoquer la mise à Faux du status de rafraîchissement s'il est Vrai et donc générer

l'envoi du message Faux (*put_F*) du module Rafraîchissement vers le module Indication réception synchronisation pour demander à ce dernier de procéder à la mise à jour du buffer liaison.

- l'échange *put_F* entre le module **Rafraîchissement synchrone** et le module **Indication réception synchronisation** résulte également, dans l'hypothèse où la synchronisation n'est pas arrivée, de la fin de la temporisation du status (ce qui provoque la mise à Faux du status).

2.3.1.2 Entité Application EA-C

Les échanges sont représentés sur la figure 8. Les échanges de services se comprennent aisément, si on a bien vu les échanges de service relatifs à l'entité EA-P (l'élément important à remarquer est que la primitive de confirmation de lecture `a_readloc.cnf(val,raf,pro)` renvoie, au PU-C, les valeurs *val* de la variable et *raf* de son rafraîchissement, reçues du producteur via le réseau, ainsi que la promptitude *pro* obtenue du module **Promptitude synchrone** suite à une requête de lecture par le message *lit*). La réception de la synchronisation (respectivement de la variable) est indiquée au module de promptitude par l'envoi du message interne *syn* (respectivement du message *var*).

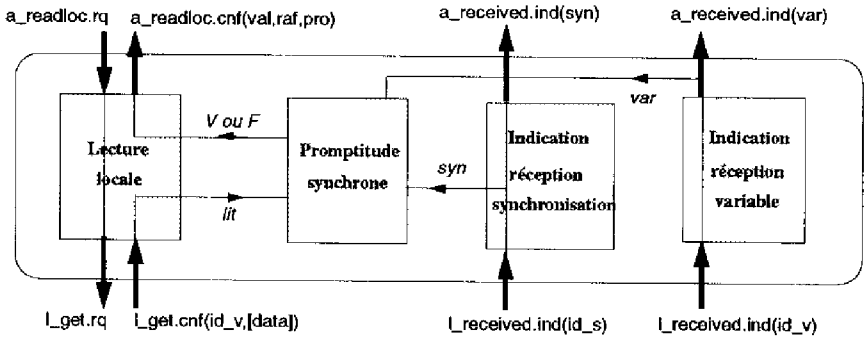

Figure 8: vue des échanges au niveau de l'EA-C

2.3.1.3 Remarque

Notons que compte tenu que les requêtes de services d'écriture locale `a_writeloc.rq(var,val)` (respectivement de lecture locale `a_readloc.rq(var)`), émises par les processus utilisateurs, ne peuvent pas être enchaînées, le nom de la variable n'est jamais contenu dans les primitives de services de confirmation d'écriture locale `a_writeloc.cnf` à l'interface avec le PU-P (respectivement de lecture locale `a_readloc.cnf(val,raf,pro)` à l'interface avec le PU-C).

2.3.2 Vue globale

La vue globale des échanges est représentée sur la figure 9. Les échanges de services découlent de la présentation générale du paragraphe 2.3.1. Les échanges de protocole font apparaître :

- d'une part, les échanges de trames `id_dat(id_v)` et `rp_dat([data])`) pour le transfert des données [data] relatives à la variable *var*. Le BA diffuse une trame requête `id_dat(id_v)` contenant l'identificateur de la variable *var* (**a**). Cette trame est alors reçue : par la station unique responsable de sa production (**b**) puis par la station consommatrice de cette variable (**c**). Au bout du temps de retournement TR de la station productrice, celle-ci émet sur le médium la trame réponse `rp_dat([data])` (**d**) où [*data*] correspond aux données $[val, raf]$ contenues dans le buffer de couche liaison de l'identifieur de la variable demandée. Cette émission sur le médium déclenche le service d'indication d'émission de la variable aux interfaces EL-P/EA-P (**d'**) et EA-P/PU-P (**d"**). Lorsque la trame réponse est reçue par la station consommatrice (**e**), les données $[val, raf]$ sont stockées dans le buffer liaison du consommateur. Cette réception déclenche le service d'indication de réception de la variable aux interfaces EL-C/EA-C (**e'**) et EA-C/PU-C (**e"**).

- d'autre part, les échanges de trames `id_dat(id_s)` et `rp_dat([ ])` pour le transfert des synchronisations. La notation [ ] signifie que le champ données de la trame est vide. Le BA diffuse la trame requête `id_dat(id_s)` (**A**) contenant l'identifieur *id_s* de la variable de synchronisation *syn*. Compte tenu que l'on ne représente pas ici le site responsable de la production de la trame réponse de synchronisation, on considère uniquement la réception de la trame `rp_dat([ ])` (qui ne contient pas de données), par la station productrice (**D**) et par la station consommatrice (**E**) pour le déclenchement de leur service respectif d'indication de réception de synchronisation ((**D'**) et (**D"**), côté station productrice et (**E'**), (**E"**) côté station consommatrice.

3 Modélisation

Nous donnons tout d'abord les modèles Réseaux de Petri des différentes entités (utilisateur, application, liaison, médium) en distinguant les deux types de transition : les transitions immédiates (représentées en trait fin) et les transitions temporisées (représentées en trait gras).
Après la présentation de ces modèles et du modèle global, nous donnons les spécifications temporelles qui sont considérées pour les transitions temporisées afin d'effectuer l'analyse comportementale de la couche application.

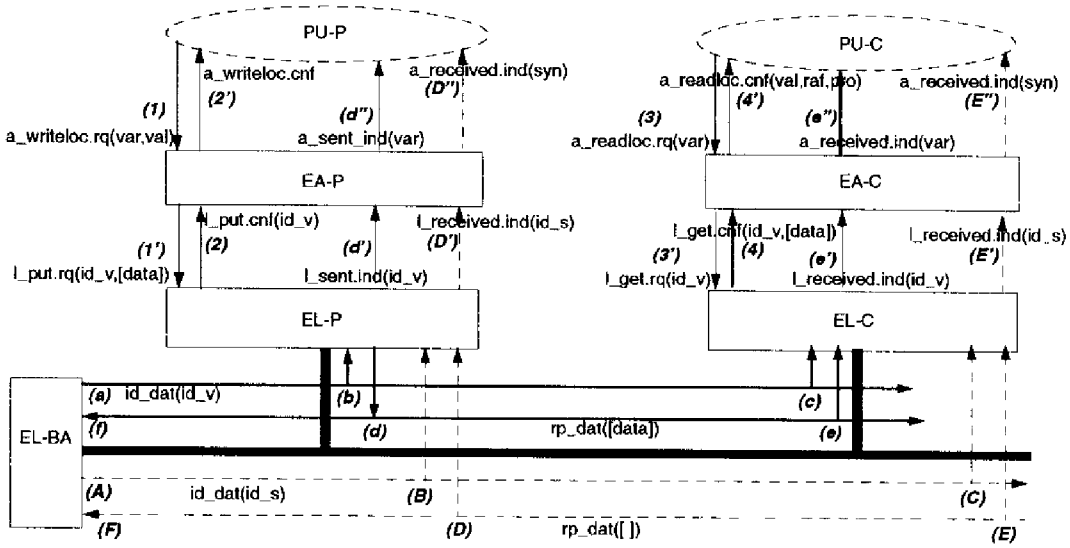

Figure 9: vue globale des échanges sur le réseau FIP

3.1 Modèles des Processus Utilisateurs

Nous considérons des processus utilisateurs très simples avec des mécanismes minimaux et qui peuvent donc être représentés par un module élémentaire.
Le PU-P représente le comportement abstrait d'un Processus Utilisateur Producteur qui alterne entre un état de `calcul` et un état de `repos` après avoir procédé à l'écriture de la valeur de la variable dans l'EA-P et attendu la confirmation relative dans l'état `attend`. De plus, nous supposons que le PU-P peut ne pas pouvoir effectuer cette écriture et passer directement à l'état de repos (au lieu d'envoyer l'écriture) ; ceci correspond à un comportement anormal, que nous appelons *perte d'écriture*.
Le PU-C représente le comportement abstrait d'un Processus Utilisateur Consommateur qui alterne entre l'état `calcule` et l'état `repos` où, au bout d'un certain temps, il procède à une demande de lecture à l'EA-C et attend la confirmation de cette lecture par l'EA-C.

Les modèles *PU-P* et *PU-C* sont représentés sur les figures 10 et 11.

Module *PU-P*

A l'initialisation, le module est dans l'état `calcule` et, au bout du temps $\mathbf{T}_{écrit}$ (tir de `t1`), il écrit la nouvelle valeur *val* de la variable *var* (dans l'EA-P) par l'envoi de la primitive de service `!a_writeloc.rq(var,val)`. Il passe alors dans l'état `attend`. Lorsqu'il reçoit la primitive de confirmation d'écriture `?a_writeloc.cnf` de l'EA-P, il passe (tir de `t2`) dans l'état `repos`. Au bout du temps $\mathbf{T}_p$ (tir de `t3`), il revient dans l'état `calcule`. La période P_{prod} du comportement du PU-P est donc fixée par : $P_{prod} = T_{écrit} + T_p$. Si le PU-P ne peut pas écrire/fournir la variable à l'EA-P, une perte d'écriture est représentée par la transition de perte `t4` qui amène directement le PU-P de l'état `calcule` à l'état `repos`.

Module *PU-C*

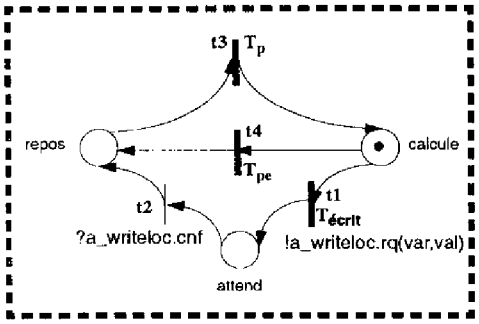

Figure 10: PU-P

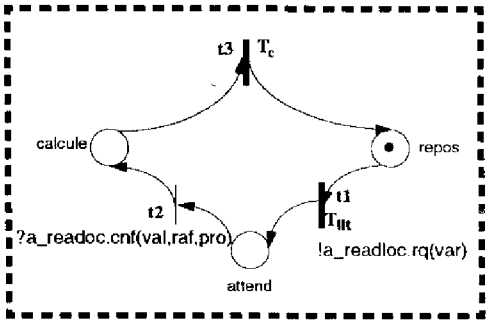

Figure 11: PU-C

A l'initialisation, le module est dans l'état `repos` et au bout du temps $\mathbf{T}_{lit}$ (tir de **t1**), il envoie la primitive de lecture `!a_readloc.rq(var)` vers l'EA-C et passe alors dans l'état `attend`. Lorsque la primitive de confirmation de lecture locale arrive `?a_readloc.cnf(val,raf,pro)`, il passe (tir de **t2**) dans l'état `calcule` où il effectue un traitement. Au bout du temps $\mathbf{T}_c$ (tir de **t3**), il revient dans l'état `repos`. La période P_{cons} du comportement du PU-C est fixée par : $P_{cons} = T_{lit} + T_c$.

3.2 Modèles des Entités Application Producteur et Consommateur

Les Entités Application Producteur et Consommateur sont constituées de plusieurs modules élémentaires et donc la méthodologie développée au chapitre 2 est utilisée (modèles des modules élémentaires - composition des modules élémentaires par fusion des places dupliquées et/ou de transitions - analyses des modules élémentaires et des modules obtenus par composition). Nous l'appliquons uniquement pour l'Entité Application Producteur (EA-P). Nous donnons ensuite directement le modèle de l'Entité Application Consommateur (EA-C).

3.2.1 Modélisation de l'EA-P

3.2.1.1 Les modèles des modules élémentaires

L'EA-P comprend les quatre modules élémentaires (cf. figure 1) : **Écriture locale** (appelé **[écrit]**), **Indication réception synchronisation** (appelé **[?s]**), **Indication émission variable** (appelé **[!v]**), **Rafraîchissement synchrone** (appelé **[raf]**). Les modèles des modules élémentaires **[écrit]**, **[raf]**, **[?s]** et **[!v]** sont représentés respectivement sur les figures 12, 13, 14 et 15 et explicités ci-dessous. Les places dupliquées importées sont représentées par des places hachurées et leur nom est précédé du nom du module auquel elles appartiennent.

Module *EA-P[écrit]*

Présentation

A l'initialisation, le module est dans la place `att_ecrit`. Lorsqu'il reçoit une requête d'écriture (`?a_writeloc.rq(var,val)`), il envoie le message *écrit* au module **[raf]**, afin de mettre à jour le status de rafraîchissement en fonction de l'état de la temporisation associée (tmp_raf), tous deux importés par les places dupliquées. Si le status est :
- Faux ($\neg raf[Vrai]$) et la temporisation est Off ($\neg tmp\_raf(On)$), alors il n'y a pas d'évolution du status ($?F$) et la requête d'écriture liaison `!l_put.rq(id_v,[val,F])` est envoyée à l'EL-P (tir de **t1**) ;
- Faux ($raf[Faux]$) et la temporisation est On ($\neg tmp\_raf(Off)$), alors le status passe à Vrai ($?V$) et la requête d'écriture liaison `!l_put.rq(id_v,[val,V])` est envoyée à l'EL-P (tir de **t2**) ;
- Vrai ($\neg raf[Faux]$), alors il le reste ($?V$) et la requête d'écriture liaison `!l_put.rq(id_v,[val,V])` est envoyée à l'EL-P (tir de **t3**).

Suite à l'écriture liaison, le module est dans l'état `att_écrit_cnf` et, dès la réception de la confirmation d'écriture liaison (`?l_put.cnf(id_v)`) de l'EL-P, l'écriture est également confirmée au PU-P (`!a_writeloc.cnf`) (tir de **t4**).

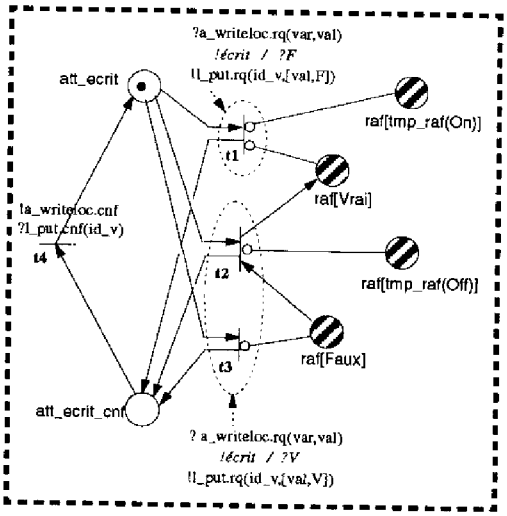

Figure 12: modèle du module **EA-P[écrit]**

Analyse

L'analyse de ce modèle, en supprimant l'ensemble des places dupliquées et en ne considérant pas les messages sur les transitions représentant les interactions avec d'autres modules, montre que, à partir de l'état initial, le système peut passer dans l'état `att_ecrit_cnf` en tirant au choix une des transitions **t1**, **t2** ou **t3**. A partir de cet état, le module se réinitialise en tirant **t4**. Le graphe d'états probabilisé est borné, sans état puits.

Module *EA-P[raf]*

Présentation

A l'initialisation, le status est dans l'état `Faux` et la temporisation `tmp_raf` est dans l'état `Off`. Les changements d'états du status sont provoqués soit :

- suite à la réception du message interne *écrit* en provenance du module **[écrit]** indiquant une requête d'écriture :
  - si la temporisation n'est pas active ($\neg tmp\_raf(On)$) et le status Faux ($\neg Vrai$), alors il le reste et la valeur du status Faux est retournée ($!F$ par le tir de **t1**) ;
  - si la temporisation est active ($\neg tmp\_raf(Off)$), alors le status Faux passe à Vrai et sa nouvelle valeur est envoyée ($!V$ par le tir de **t2**) ;
  - si le status est déjà Vrai ($\neg Faux$), alors il le reste et sa valeur est envoyée ($!V$ par le tir de **t3**) ;

- suite à la réception du message interne d'indication de synchronisation (*?syn*) en provenance du module **[?s]** :
  - si la temporisation est `Off`, alors elle est initialisée (tir de **t9**) ;
  - si la temporisation est `On`, alors elle est réinitialisée et : soit le status est Faux ($\neg Vrai$) et il le reste (tir de **t8**) ; soit le status est Vrai et il passe à Faux (tir de **t7**), ce qui provoque l'envoi du message *put_F* pour demander une mise à jour du buffer liaison au module **[?s]** ;

- par l'évènement interne de fin de temporisation du rafraîchissement au bout de la durée $\mathbf{T}_{raf}$ (tir de **t6**) (spécifiée identique à la période de synchronisation afin de détecter une perte de synchronisation). La temporisation n'étant plus active ($\neg tmp\_raf(On)$), la mise à Faux du status, qui était Vrai, provoque (tir de **t5**) une demande de mise à Faux du status dans le buffer liaison au module **[?s]** (*put_F*).

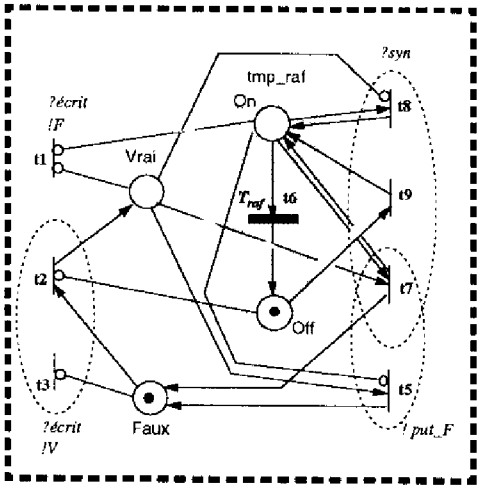

Figure 13: modèle du module **EA-P[raf]**

Analyse

L'analyse de ce module, en supprimant ses interactions avec les autres modules élémentaires, montre un comportement qui évolue (par exemple, tir de la séquence **t9**, **t2**, **t7**, etc.). Le

graphe d'états probabilisé est borné et sans état puits.

Module *EA-P[?s]*

Présentation

A l'initialisation, le module est dans l'état `att_syn`. Lorsqu'une indication de réception de synchronisation est reçue de l'EL-P (`?l_received.ind(id_s)`), l'indication de synchronisation est émise vers le PU-P (`!a_received.ind(syn)`) ; le message interne *!syn* est alors émis, ce qui va agir sur le module **[raf]** en initialisant ou réinitialisant la temporisation du module `[raf]` :
- si elle n'est pas en cours (tir de **t9**) ;
- si elle est active et le status Faux ($\neg raf[Vrai]$) (tir de **t8**) ;
- si elle est active et le status Vrai, il est mis à Faux (tir de **t7**) et lorsque le message de mise à Faux du status (*put_F*) est reçu du module **[raf]**, l'écriture du buffer liaison (`!l_put.rq(id_v,[val,F])`) est effectuée et le module passe dans l'état `att_cnf`.

De plus, si une mise à jour du status est demandée par le module **[raf]** (*?put_F*), à la fin de la synchronisation de rafraîchissement, une écriture du buffer liaison est effectuée (`!l_put.rq(id_v,[val,F])`) (tir de **t5**), ce qui amène le module dans l'état `att_cnf`.
A partir de la place `att_cnf`, lorsque la primitive de confirmation d'écriture liaison est reçue de l'EL-P (`?l_put.cnf(id_v)`), le module retourne dans l'état initial `att_syn` (tir de **t10**).

Analyse

L'analyse de ce module est effectuée en supprimant ses interactions. A partir de l'état initial, une des transitions **t7**, **t8** ou **t9** peut être tirée au choix, ce qui amène le module dans l'état `att_cnf`. Le graphe d'état est borné, sans état puits et le modèle est réinitialisable.

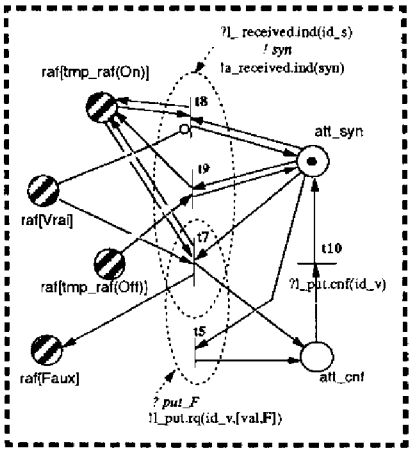

Figure 14: modèle du module **EA-P[?s]**

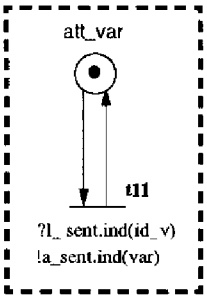

Figure 15: modèle du module **EA-P[!v]**

Module *EA-P[!v]*

Présentation

A l'initialisation, le module est dans l'état `att_var`. Lorsque l'indication d'émission sur le médium de la variable *var* (`?l_sent.ind(id_v)`) est reçue de l'EL-P (liée au service de transfert de buffer périodique géré par le BA), celle-ci provoque l'envoi de l'indication d'émission (`!a_sent.ind(var)`) vers le PU-P (tir de `t11`). Ce module ne contient qu'une seule place.

Analyse
L'analyse de ce module, sans considérer ses interactions, est évidente ; c'est une boucle qui donne un graphe possédant les propriétés : réinitialisable, borné et sans état puits.

3.2.1.2 Modèle complet de l'EA-P

Présentation
La composition des modèles **[écrit]**, **[raf]**, **[?s]** et **[!v]** est effectuée en fusionnant les transitions en rendez-vous et les places dupliquées ; le modèle **EA-P** obtenu est représenté sur la figure 16.

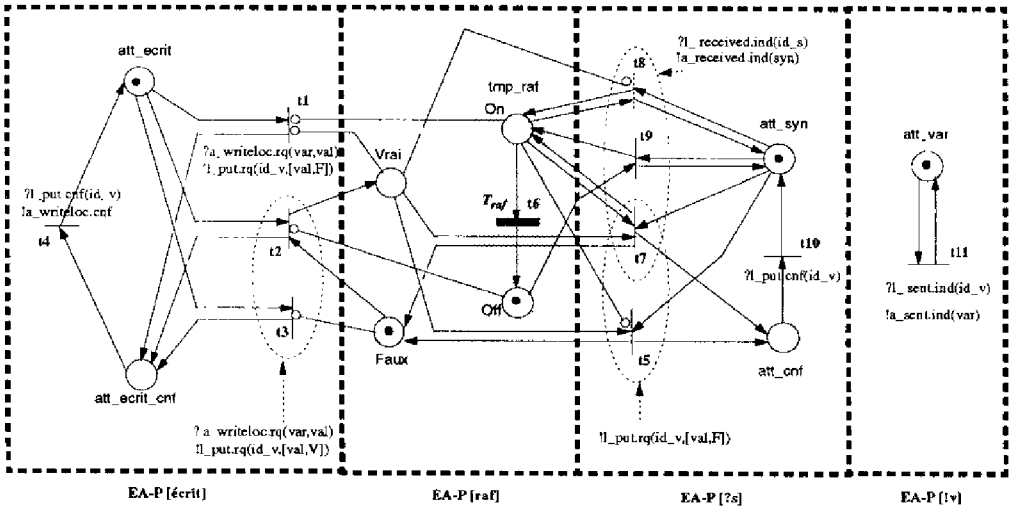

Figure 16: modèle de l'EA-P

Analyse
L'analyse du modèle complet de l'EA-P sans considérer ses interactions donne un graphe borné (17 états), sans état puits. Les propriétés générales et spécifiques de l'EA-P (en le connectant à un modèle réduit de l'environnement générant les écritures du PU-P et les indications de réception de synchronisation, ainsi que d'émission de la variable) ont été analysées dans [BJSA94].

3.2.2 Modèle de l'EA-C

L'EA-C comprend les quatre modules élémentaires (cf. figure 3) : **Lecture locale** (appelé **[lit]**), **Indication réception synchronisation** (appelé **[?s]**), **Indication réception va-**

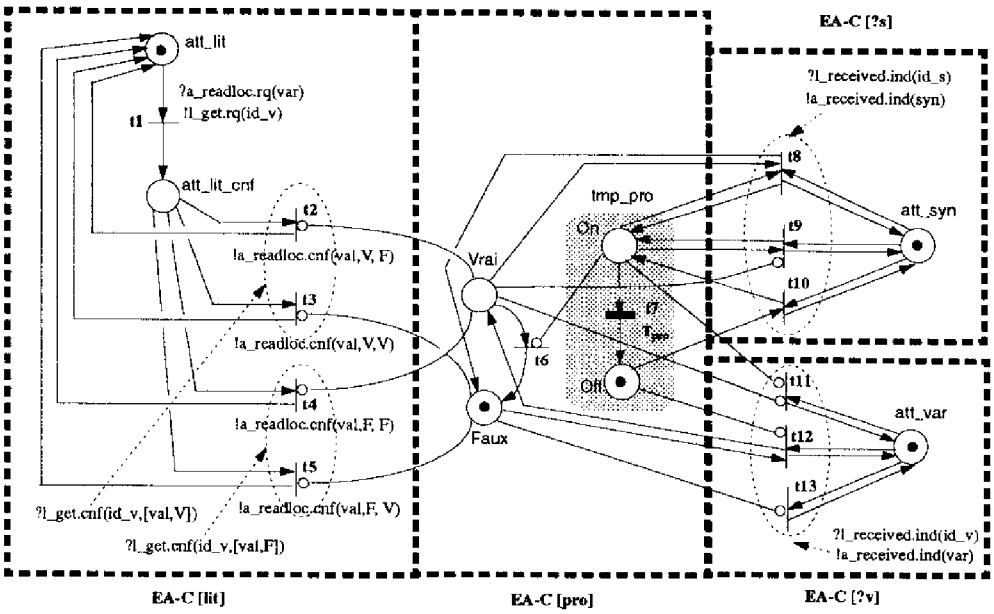

Figure 17: modèles de l'EA-C

riable (appelé **[?v]**), **Promptitude synchrone** (appelé **[pro]**). Le modèle RdP de l'EA-C est représenté sur la figure 17.

Module *EA-C[lit]*

A l'initialisation, le module est dans la place `att_lit`. Lorsqu'une demande de lecture locale de *var* est reçue du PU-C (`?a_readloc.rq(var)`), la demande de lecture du buffer liaison est immédiatement envoyée à l'EL-C (`!l_get.rq(id_v)`) (tir de **t1**) et la place `att_lit_cnf` est marquée.

Lorsque la confirmation de lecture du buffer liaison est reçue de l'EL-C telle que :

- `?l_get.cnf(id_v,[val,V])` où le status de rafraîchissement est Vrai :
  - si le status de promptitude est `Faux`, alors la primitive de confirmation de lecture `!a_readloc.cnf(val,V,F)` est renvoyée au PU-C (tir de **t2**) ;
  - si le status de promptitude est `Vrai`, alors la primitive de confirmation de lecture `!a_readloc.cnf(val,V,V)` est renvoyée au PU-C (tir de **t3**) ;
- `?l_get.cnf(id_v,[val,F])` où le status de rafraîchissement est Faux :
  - si le status de promptitude est `Faux`, alors la primitive de confirmation de lecture `!a_readloc.cnf(val,F,F)` est renvoyée au PU-C (tir de **t4**) ;
  - si le status de promptitude est `Vrai`, alors la primitive de confirmation de lecture `!a_readloc.cnf(val,F,V)` est renvoyée au PU-C (tir de **t5**).

Module *EA-C[?s]*

A l'initialisation, il est dans la place `att_syn` en attente d'indication de réception de la variable de synchronisation de l'EL-C (`?l_received.ind(id_s)`), ce qui provoque l'émission de la primitive d'indication de réception de synchronisation vers le PU-C

(`!a_received.ind(syn)`) dans le cas où :
- la temporisation est en cours et le status est `Vrai` ; ce dernier passe alors à `Faux` (tir de **t8**) ;
- la temporisation est en cours et le status `Faux` ($\neg\ pro(Vrai)$), il n'y a alors pas d'actions sur le status (tir de **t9**) ;

la temporisation est désactivée ; elle est alors initialisée (tir de **t10**) et il n'y a pas d'actions sur le status (il est déjà Faux).

Module *EA-C[?v]*
A l'initialisation, il attend dans la place `att_var` les indications de réception de la variable *var* de l'EL-C (`?l_received.ind(id_v)`), ce qui provoque l'émission de l'indication de réception vers le PU-C (`!a_received.ind(var)`) et, :
- si la temporisation du status est `Off` ($\neg\ pro(tmp\_pro(On))$) et le status est Faux ($\neg\ Vrai$), il n'y a pas d'évolution du status (tir de **t11**) ;
- si la temporisation est `On` ($\neg pro(tmp\_pro(Off))$) et le status `Faux`, ce dernier passe à `Vrai` (tir de **t12**) ;
- si le status est déjà `Vrai` ($\neg\ pro(Faux)$), il le reste (tir de **t13**).

Module *EA-C[pro]*
A l'initialisation, le statut est dans l'état `Faux` et la temporisation est dans l'état `tmp_pro(Off)`. Comme déjà présenté, les changements d'états du status de promptitude sont provoqués soit, :
- par les réceptions de synchronisation (sous le contrôle du module [?s]) ;
- par les réceptions de la variable *var* (sous le contrôle du module [?v]) ;
- par l'évènement interne de fin de temporisation de promptitude (tir de **t7**) au bout de la durée $\mathbf{T}_{pro}$ (spécifiée approximativement égale à la période de synchronisation), ce qui provoque la mise à jour du status de Vrai à Faux (tir de **t6**).

3.3 Modèles des Entités Liaison

Notre objectif n'est pas ici de faire une modélisation exhaustive des Entités Liaison, mais de représenter les mécanismes strictement nécessaires à l'étude de la mise en œuvre des échanges pour la coopération synchrone entre les deux entités EA-P et EA-C de la couche application.

En outre, comme nous ne représentons pas la station productrice de la variable de synchronisation (qui doit diffuser la trame réponse `rp_dat[ ]`, suite à la réception de la trame `id_dat(id_s)`), nous considérons que l'Arbitre de Bus émet directement la trame `rp_dat[ ]` (et n'émet donc pas la trame `id_dat(id_s)`).

Les modèles EL-P et EL-C, qui comprennent plusieurs modules élémentaires, sont donnés directement après composition par fusion des places dupliquées et des transitions en rendez-vous. Les modèles de l'EL-BA et du médium comprennent un seul module élémentaire.

3.3.1 Modèle de l'EL-BA

Le modèle de l'EL-BA (figure 18) représente l'envoi périodique, tout d'abord, de la trame réponse de synchronisation `rp_dat([])` puis, de la trame requête de la variable de données `id_dat(id_v)`.

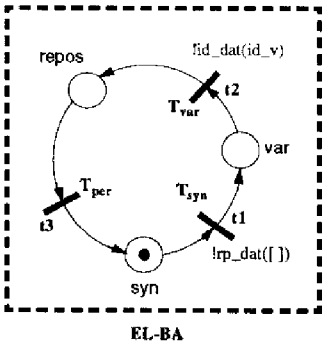

Figure 18: modèle de l'EL-BA

Module *EL-BA*

A l'initialisation, le module est dans l'état **syn**, prêt à émettre la trame réponse de synchronisation (avant la trame requête de la variable donnée, afin d'activer les temporisations et de raccourcir le régime transitoire). Au bout du temps T_{syn}, la trame `!rp_dat([ ])` (tir de **t1**) est envoyée sur le médium et le module passe dans l'état **var**. Au bout du temps T_{var} (déphasage entre l'émission de la trame réponse de la variable de synchronisation et la trame requête de transfert de la variable *var*), la trame `!id_dat(id_v)` (tir de **t2**) est envoyée sur le médium et le module passe dans l'état **repos**. Dans cet état, il attend la fin du temps T_{per} (tir de **t3**) qui correspond à la durée d'attente pour garantir la périodicité du cycle de diffusion des variables *var* et *syn* sur le bus ($P_{bus}(var) = T_{syn} + T_{var} + T_{per}$).

3.3.2 Modèle de l'EL-P

L'EL-P comprend (cf. figure 4) quatres modules élémentaires dont trois sont relatifs aux services périodiques, qui traduisent des fonctions du même type que celles de l'EA-P et sont donc notés **[écrit]**, **[?s]** et **[!v]**, et le module relatif au buffer de stockage (appelé **[Bprod]**). La figure 19 représente le modèle de l'EL-P.

Module *EL-P[Bprod]*

Il contient les places `[val,F]` (marquée à l'initialisation) et `[val,V]` qui permettent de distinguer la valeur du status de rafraîchissement fourni par l'EA-P.
Les changements d'états sont gérés par le module **[écrit]**.

Module *EL-P[écrit]*

Il ne comprend pas de place ; il est réduit aux transitions en interaction avec le buffer **Bprod**.
Lorsqu'une requête d'écriture est reçue de l'EA-P :

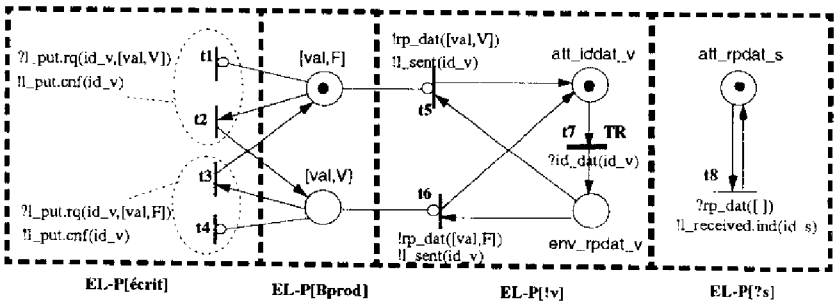

Figure 19: modèle de l'EL-P

- avec le status Vrai (`?l_put.rq(id_v,[val,V])`) :
  - si le buffer contient la valeur Vrai ($\neg[Bprod([val, F])]$), alors il n'y a pas d'évolutions (tir de **t1**) ;
  - si le buffer contient la valeur Faux, il passe à Vrai (tir de **t2**) ;
- avec le status Faux (`?l_put.rq(id_v,[val,F])`) :
  - si le buffer contient la valeur Vrai, alors il passe à Faux (tir de **t3**) ;
  - si le buffer contient la valeur Faux ($\neg[Bprod([val, V])]$), alors il n'y a pas d'évolution (tir de **t4**).

Dans tous les cas, la confirmation d'écriture est renvoyée à l'EA-P (`!l_put.cnf(id_v)`).

Module *EL-P[!v]*

A l'initialisation, il est dans l'état `att_iddat_v`. Lorsque la requête de transfert de buffer sur le médium est reçue (`?id_dat(id_v)`), alors au bout du temps de retournement TR, il passe dans l'état `env_rpdat_v` (tir de **t7**) où il est prêt à émettre sur le médium la trame réponse avec :

soit, le status Vrai (`!rp_dat([val,V])`) (tir de **t5**) si le module **[Bprod]** contient ces données ($\neg[Bprod([val, F])]$) ;
– soit, le status Faux (`!rp_dat([val,F])`) (tir de **t6**) si le module **[Bprod]** contient ces données ($\neg[Bprod([val, V])]$).

Module *EL-P[?s]*

A l'initialisation, il est dans la place unique `att_rpdat_s` et lorsqu'il reçoit la trame réponse de synchronisation `?rp_dat([ ])`, il reste dans cette place et envoie l'indication de réception de synchronisation à l'EA-P (`!l_received.ind(id_s)` par le tir de **t8**).

3.3.3 Modèle de l'EL-C

L'EL-C comprend (cf. figure 5) trois modules élémentaires relatifs aux services périodiques, qui réalisent les même fonctions que celles de l'EA-C, les modules sont donc notés **[lit]**, **[?v]**

et [?s]. Le module relatif au buffer de stockage est appelé [**Bcons**]. La figure 20 représente les modèles de l'EL-C.

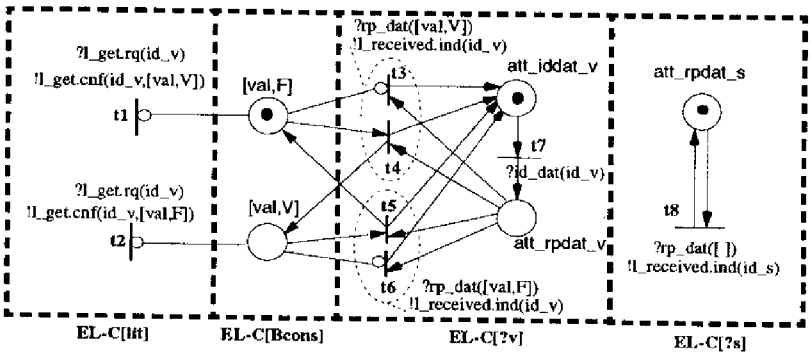

Figure 20: modèle de l'EL-C

Module *EL-C[Bcons]*

Il contient les places `[val,F]` (marquée à l'initialisation) et `[val,V]` qui permettent de distinguer la valeur du status de rafraîchissement reçu. Les changements d'états sont gérés par le module [**?v**].

Module *EL-C[?v]*

A l'initialisation, il est dans la place `att_iddat_v`. Lorsque la trame requête `?id_dat(id_v)` est reçue (tir de **t7**), il passe dans l'état `att_rpdat_v`. A partir de cet état, le stockage est effectué dans le module [**Bcons**] en fonction des données reçues [*data*] dans la trame réponse :

- soit, avec le status Vrai (`?rp_dat([val,V])`) :
  - si le buffer contient la valeur Vrai ($\neg[Bprod([val, F])]$), alors il n'y pas de d'évolution (tir de **t3**) ;
  - si le buffer contient la valeur Faux, alors il passe à Vrai (tir de **t4**) ;
- soit, avec le status Faux (`?rp_dat([val,F])`) ;
  - si le buffer contient la valeur Vrai, alors il passe à Faux (tir de **t5**) ;
  - si le buffer contient la valeur Faux ($\neg[Bcons([val, V])]$), alors il n'y a pas d'évolution (tir de **t6**) ;

  Dans tous les cas, la primitive d'indication de réception de la variable est envoyée à l'EA-C (`!l_received.ind(id_v)`) et le module revient dans l'état `att_iddat_v`.

Module *EL-C[lit]*

Il ne contient pas de place ; il est réduit aux transitions en interaction avec le module [**Bcons**] pour effectuer les accès en lecture sur requêtes de l'EA-P. Lorsqu'une requête de lecture est reçue de l'EA-C (`?l_get.rq(id_v)`), il renvoie la primitive de confirmation à l'EA-C en fonction des données contenues dans le buffer, c'est-à-dire :

- soit, avec le status Vrai (`!l_get.cnf(id_v,[val,V])`) ($\neg[Bcons([val,F])]$) (tir de **t1**) ;
  soit, avec le status Faux (`!l_get.cnf(id_v,[val,F])`) ($\neg[Bprod([val,V])]$) (tir de **t2**).

Module *EL-C[?s]*
A l'initialisation, il est dans la place unique `att_rpdat_s` et lorsqu'il reçoit la trame réponse de synchronisation `?rp_dat([ ])`, il reste dans cette place et envoie l'indication de réception de synchronisation à l'EA-C (`!l_received.ind(id_s)` par le tir de **t8**) .

3.3.4 Modèle du médium

Le modèle du médium utilisé ici (figure 21) représente une abstraction des comportements du contrôle d'accès des entités liaison pour les échanges des trames `id_dat(id_v)`, `rp_dat([data])` et `rp_dat([])`.

Nous supposons que le site producteur est à égale distance du site arbitre et du site consommateur. Nous appelons :
- τ le temps de propagation entre l'arbitre et le producteur et, entre le producteur et le consommateur ;
- T_1 et T_2 les durées respectives des trames `id_dat` et `rp_dat`.

Enfin, nous considérons que l'on peut avoir une perte de trame de synchronisation `rp_dat([ ])` entre le site producteur et le site consommateur, ce que nous appelons une *perte de synchronisation* (ceci permet d'analyser les mécanismes de validité temporelle des données au niveau du consommateur). Les autres transferts se déroulent correctement.

Module *médium*
Il attend, sur la transition d'entrée **t1**, la réception de la trame `?id_dat(id_v)` en provenance de l'EL-BA puis, il passe dans l'état **v1** représentant le transfert de cette trame vers l'EL-P. Au bout du temps de transfert (temps de propagation et temps d'émission)($\tau + T_1$) de cette trame, celle-ci arrive à l'EL-P. Lorsque la trame réponse est reçue de l'EL-P :
- avec le status Vrai `?rp_dat([val,V])` (tir de la transition d'entrée **t5**), le module passe dans la place **vV1** qui représente le transfert de cette trame vers l'EL-C ;
- avec le status Faux `?rp_dat([val,F])` (tir de la transition d'entrée **t6**), le module passe dans la place **vF1** qui représente la transmission de cette trame vers l'EL-C.

Au bout du temps de transfert (propagation et émission) ($\tau + T_2$) de la trame réponse de l'EL-P vers l'EL-C :
- avec le status Vrai, la trame réponse `rp_dat([val,V])` est envoyée à l'EL-C (tir de **t8**) ;
- avec le status Faux, la trame réponse `rp_dat([val,F])` est envoyée à l'EL-C (tir de **t9**).

Il attend, sur la transition d'entrée **t2**, la réception de la trame `?rp_dat([ ])` en provenance de l'EL-BA puis, il passe dans l'état **s1** représentant la propagation de cette trame du BA vers l'EL-P. Au bout du temps de propagation τ, cette trame réponse est transmise vers l'EL-P et vers l'EL-C (tir de **t4**). La place **s3** représente l'émission de cette trame vers l'EL-P ; au bout du temps d'émission (T_2), elle est envoyée à l'EL-P (`!rp_dat([ ])`) (tir de **t11**). La place **s2** représente le transfert (propagation et émission) de cette trame vers

l'EL-C ; au bout du temps de transfert (propagation et émission) ($\tau + T_2$), cette trame est envoyée vers l'EL-C (`!rp_dat([ ])`) (tir de **t10**).

La perte de la trame réponse de synchronisation sur le médium entre l'EL-P et l'EL-C est représentée par la transition **t7** à partir de l'état **s2**.

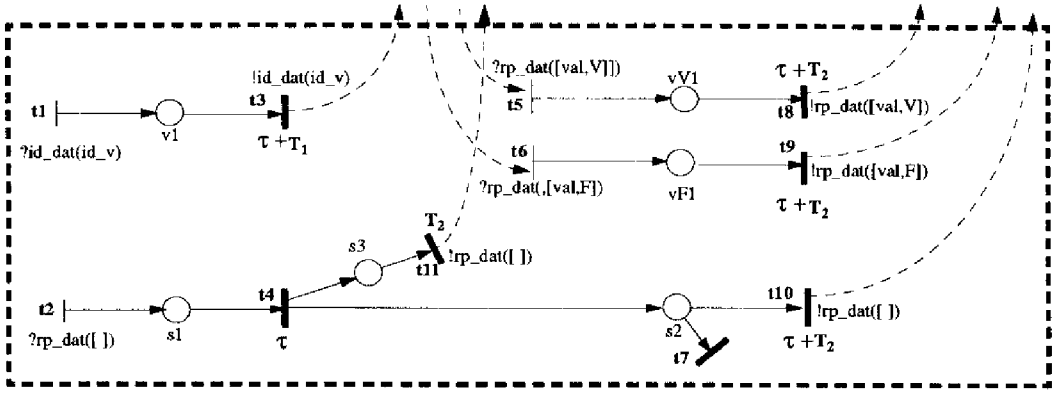

Figure 21: modèle du médium

3.4 Modèle global

Le modèle complet serait obtenu en composant l'ensemble des modèles (`PU-P, EA-P, EL-P, EL-BA, médium, EL-C, EA-C, PU-C`) par l'introduction de places de communication entre les modèles de modules de niveaux adjacents qui ont des interactions. Cependant, pour simplifier l'étude, nous proposons de réduire certains modèles afin de limiter le nombre de places et transitions et ainsi, faciliter l'analyse en diminuant la taille du graphe d'accessibilité.

3.4.1 Réduction de modèles

Les réductions effectuées portent uniquement sur les modèles de niveau liaison c'est-à-dire l'`EL-P`, l'`EL-C` et le `médium`. Elles utilisent les règles de réduction définies par Berthelot [BRV80] pour des réseaux de Petri non temporisés (places implicites - transitions séries). Ces règles peuvent être appliquées pour les réseaux de Petri que nous utilisons (on peut réduire une série de plusieurs transitions temporisées avec des distributions déterministes en une seule transition dont la distribution déterministe a pour valeur la somme des valeurs associées aux distributions déterministes initiales).

- Dans l'EL-P, le module EL-P[!v] est réduit aux transitions **t5** et **t6** ; les places `att_iddat_v` et `att_rpdat_v` sont supprimées par substitution et le temps de retournement TR, associé initialement à la transition EL-P[!v(t7)], est ajouté à la transition `médium[t3]` (car elles possèdent toutes les deux des distributions déterministes).

De même, dans le module EL-C[?v], les places `att_iddat_v` et `att_rpdat_v` sont supprimées par substitution ; il est donc réduit aux transitions **t3**, **t4**, **t5** et **t6** auxquelles est associée directement la réception de la trame `id_dat(id_v)` (initialement sur **t7**).

- Les modules EL-P[?s] et EL-C[?s] sont supprimés par réduction des places implicites `att_rpdat_s` et de leur transition **t8** respective.
- Dans le modèle du médium,
  - la transition **t1** et la place `v1` sont supprimées par substitution ; la transition **t3** qui représente le cumul du temps de propagation et la durée de transfert vers l'EL-P, ainsi que le temps de retournement de ce dernier ;
  - de même, les transitions **t5** et **t6**, ainsi que les places `vF1`, `vV1` sont supprimées par substitution ;
  - la place `s1` est supprimée par substitution et son délai τ est associé à la transition **t11** vers l'EL-P, ainsi qu'à la transition **t10** vers l'EL-C.

3.4.2 Le modèle global réduit

Le modèle global obtenu après réduction, qui sera considéré dans l'analyse, est représenté sur la figure 22 (les places grises entre les modèles des Processus Utilisateurs, Entités Application, Entités Liaison, médium et Arbitre de Bus représentent les places de communication introduites pour la composition des modèles).

3.5 Spécifications temporelles

Les valeurs importantes sont les suivantes :
- les périodes d'écriture et de lecture des processus utilisateurs et la période de mise à jour réseau qui sont égales (cf. § 3.4.2 au chapitre 3) et qui ont pour valeur $50ms$;
- les durées des temporisations associées aux status de rafraîchissement et de promptitude qui sont de $51ms$;
- le temps de propagation τ qui est de $0.5\mu s$ (distance de 100m) ;
- les durées T_1 et T_2 d'émission des trames `id_dat` et `rp_dat` qui sont respectivement de $61\mu s$ et de $77\mu s$ à la vitesse de transmission de 1 Mb/s (la durée d'une trame `rp_dat` dépend du nombre d'octets utilisé pour les données utilisateur à transférer, ici 2 octets) ;
- le temps de retournement TR du modem du site producteur qui est choisi de $20\mu s$.

Nous adoptons comme unité de temps (Ut) le dixième de milliseconde et la précision est de l'ordre du dixième de Ut. Compte tenu de ces notations, les distributions temporelles des transitions temporisées sont indiquées sur le tableau 1. Les transitions `PU-P[t4]` et `médium[t7]` permettent de fixer la probabilité d'occurrence (0.005) respectivement d'une perte d'écriture et d'une perte de synchronisation [JA91]. Toutes les transitions dont les

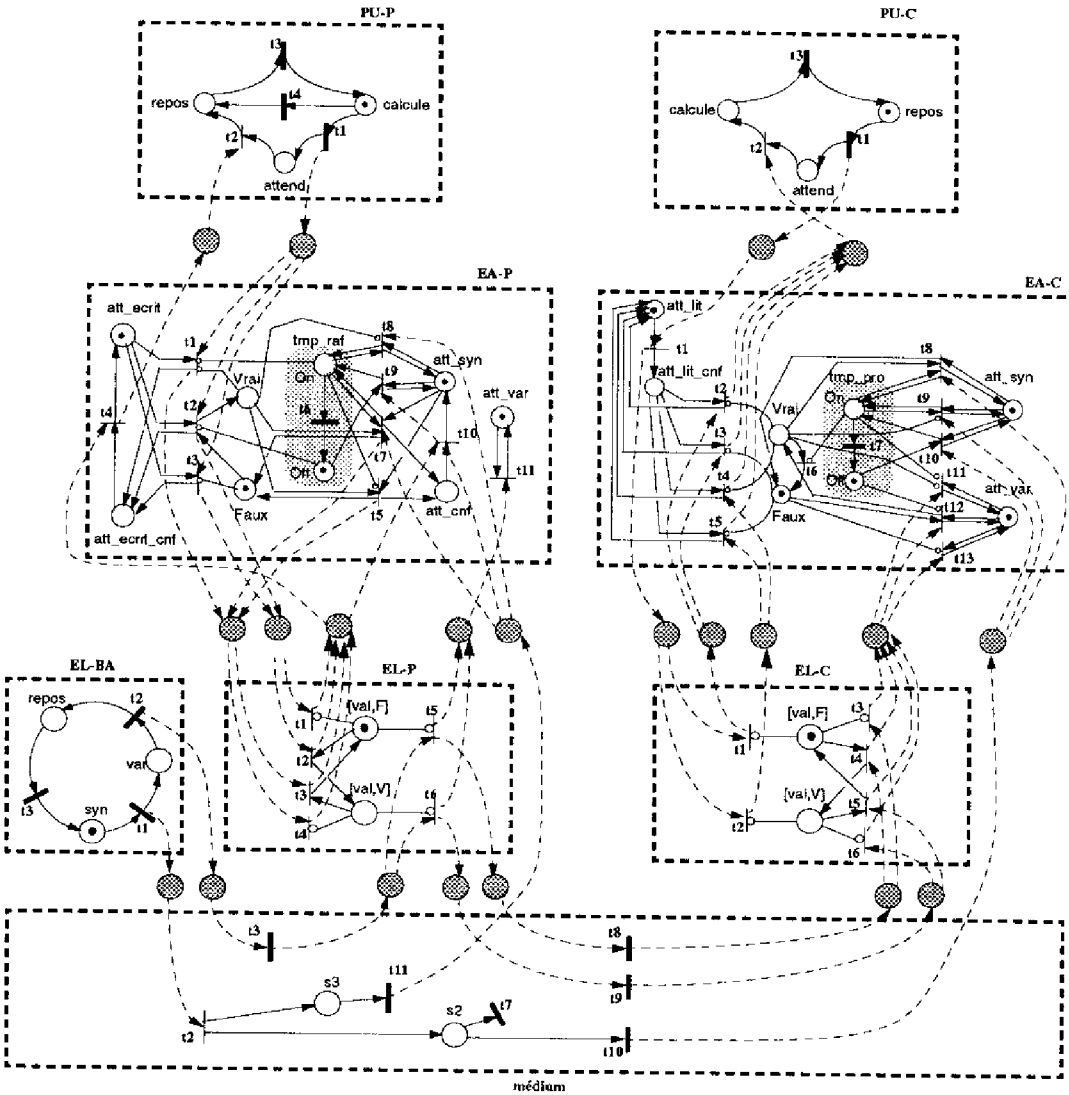

Figure 22: modèle RdP global réduit

spécifications temporelles ne sont pas citées dans le tableau sont des transitions immédiates définies par : $f(x) = \delta(x)$.

Module	Transition	Distribution temporelle $f(x)$
PU-P	t1	$\delta(x-80)$
	t3	$\delta(x-420)$
	t4	$0.01\delta(x-80)+0.99$ tel que $80 \leq x \leq 81$
PU-C	t1	$\delta(x-140)$
	t3	$\delta(x-360)$
EA-P	t6	$\delta(x-510)$
EA-C	t7	$\delta(x-510)$
EL-BA	t1	$\delta(x-50)$
	t2	$\delta(x-60)$
	t3	$\delta(x-390)$
médium	t8, t9, t11	$\delta(x-0.8)$
	t10	$\delta(x-0.9)$
	t3	$\delta(x-1)$
	t7	$0.01\delta(x-0.9)+0.99$ tel que $0.9 \leq x \leq 1.9$

Tableau 1: spécifications temporelles des transitions temporisées

4 Analyse globale

4.1 Cadre de l'analyse

L'analyse est effectuée en considérant :
- tout d'abord, il n'y a pas de perte d'écriture et/ou de synchronisation, c'est-à-dire le fonctionnement est normal ;
- ensuite, qu'il y a pertes d'écriture et/ou de synchronisation, c'est-à-dire que des fonctionnements anormaux peuvent se produire.

Les graphes d'états probabilisé, qui n'ont pas d'état puits et ont des états d'accueil, comptent :
- 49 états quand on considère uniquement le fonctionnement normal ;
- 80 états quand on considère l'hypothèse de pertes d'écriture ;
- 95 états quand on considère l'hypothèse de pertes de synchronisation ;
- 158 états quand on considère les hypothèses de pertes à la fois d'écriture et de synchronisation.

Ici, nous voulons surtout nous concentrer sur les propriétés spécifiques de la couche application et plus précisément, le service application en termes de séquences de primitives de services locaux (d'écriture, de lecture et d'indication de réception et d'émission) et des status

de rafraîchissement et de promptitude associés aux données coté consommateur.

Afin de faire cette analyse, nous considérons une vue abstraite du graphe d'états probabilisé qui est obtenue en faisant une projection (cf. §4.3.2.3 chapitre 1) sur les états atteints, au niveau de l'interface application, après la réception d'une primitive de service (venue de l'utilisateur) ou après l'envoi d'une primitive de service (vers l'utilisateur). Les états apparaissent avec le numéro du graphe d'états probabilisé. Les arcs du graphe de la vue abstraite sont étiquetés avec : les primitives de services provoquant le changement d'état dans l'état suivant ; le couple [temps moyen, probabilité] associé au changement d'état. Les couples (`Valeur_status ; Etat_temporisation`) sont associés aux états des vues abstraites avec : à gauche, le couple relatif à l'état courant du status de rafraîchissement et de sa temporisation dans l'EA-P et à droite, le couple relatif à l'état courant du status de promptitude et de sa temporisation dans l'EA-C. Nous indiquons la valeur de ces couples sur l'état initial et sur les états après changement d'une valeur.

Pour chaque cas étudié, nous effectuons tout d'abord une analyse qualitative, puis une analyse quantitative dont le but est de quantifier les propriétés qualitatives et d'évaluer également la sûreté de fonctionnement par rapport aux status de validité temporelle (nous nous situons en régime permanent et nous appliquerons la méthodologie présentée au chapitre 1 pour calculer le MTFF relativement à la *relation producteur-consommateur* : réception d'écriture et envoi de confirmation de lecture).

4.2 Fonctionnement normal

La vue abstraite, représentée sur la figure 23, nous permet d'exprimer les propriétés suivantes :

- Analyse qualitative
  - on a un régime transitoire et un régime permanent cyclique (partie grise) ; ceci résulte du fait que les temporisations sont à l'état `Off` et que les status sont à l'état `Faux` à l'initialisation ;
  - à partir de l'état 4, suite à l'envoi de la variable de synchronisation au processus utilisateur producteur (`EA-P!a_received.ind(syn)`), on peut vérifier que l'on a inévitablement la séquence de primitives qui conduit à l'état 22 où on a l'envoi, au processus utilisateur consommateur, de la confirmation de lecture de la variable reçue avec les status de rafraîchissement et de promptitude vrais (`EA-C!a_readloc.cnf(val,V,V)`) ; ceci traduit le fonctionnement correct.
  - à partir de l'état 30, on a la même séquence qui ramène à l'état 22 ;
- Analyse quantitative

  On peut quantifier les propriétés précédentes :
  - la durée du cycle du régime permanent est égale à 500 Ut (i.e. cette durée est égale à la période des cycles utilisateurs de lecture et d'écriture, ainsi que du cycle de

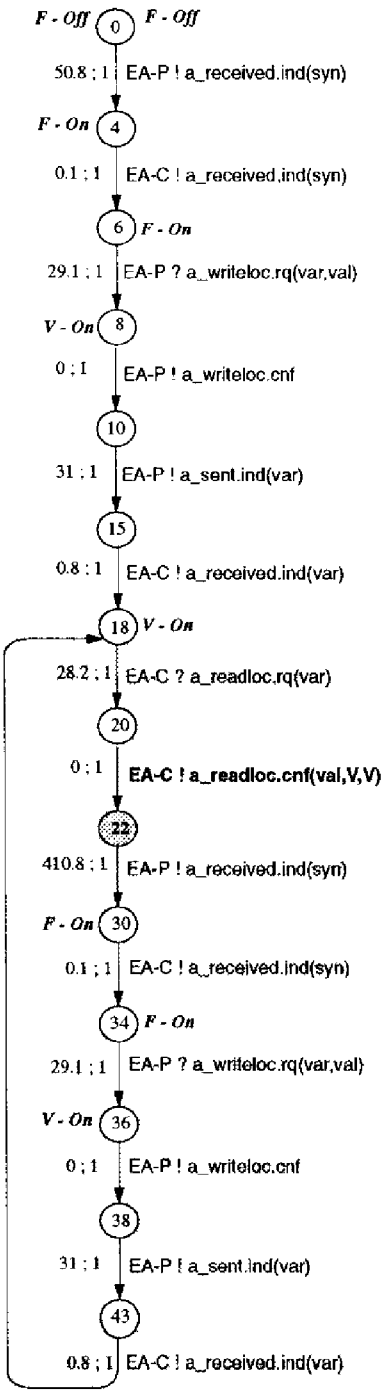

Figure 23: vue abstraite du service application

mise à jour réseau ; ce qui est logique) ;

- après l'envoi de la variable de synchronisation au processus utilisateur producteur (`EA-P!a_received.ind(syn)`), en régime transitoire (état 4) ou en régime permanent (état 30), on a, avec la probabilité 1 et après un temps de 88.2 Ut, l'envoi au Processus Utilisateur Consommateur de la confirmation de lecture avec les status de rafraîchissement et de promptitude vrais (`EA-C!a_readloc.cnf(val,V,V)`) ;
- après une réception d'écriture (`EA-P?a_writeloc.rq(var,val)`) (états 8 et 36), on a, avec la probabilité 1 et après un temps de 60 Ut, l'envoi de la confirmation de lecture (`EA-C!a_readloc.cnf(val,V,V)`) (... → 22) au processus utilisateur consommateur.

4.3 Prise en compte d'une perte d'écriture

La vue abstraite, représentée sur la figure 24 (où les transitions en pointillé représentent l'occurrence des pertes d'écriture), fait apparaître les propriétés suivantes :

- Analyse qualitative
  - on retrouve toujours le régime transitoire et le régime permanent du fonctionnement normal. Mais, on voit maintenant des cycles inhérents aux pertes d'écriture ;
  - en régime transitoire, après l'état 4 (`EA-P!a_received.ind(syn)`), on a :
    * la potentialité d'une confirmation de lecture correcte (`EA-C!a_readloc.cnf(val,V,V)`) dans l'état 22, suite à l'occurrence de la réception d'une écriture (`EA-P!a_writeloc.rq(var,val)` dans l'état 8) ;
    * la potentialité d'une confirmation de lecture avec un status de rafraîchissement faux (`EA-C!a_readloc.rq(val,F,V)`) dans l'état 56, suite à l'occurrence d'une perte d'écriture (état 74) ;
  - en régime permanent, après l'état 22 (`EA-P!a_received.ind(syn)`), on a également les potentialités de confirmation de lecture correcte (on revient à l'état 22) et de confirmation de lecture avec un status de rafraîchissement faux (on va à l'état 56 en passant par l'état 45 ;
  - notons qu'après l'état 56, on peut se resynchroniser dans le comportement normal (états 6 → 8...) ou repartir dans un fonctionnement incorrect (états 6 → 74).
- Analyse quantitative
  - Quantification des relations de potentialités précédentes : par exemple, dans l'état 30 du régime permanent normal, après `EA-P!a_received.ind(syn)` :
    * on a, avec une probabilité 0.995 et au bout d'un temps de 88.2 Ut, la confirmation de lecture vers le PU-C avec tous les status vrais (`EA-P!a_readloc.cnf(val,V,V)`) dans l'état 22 ;

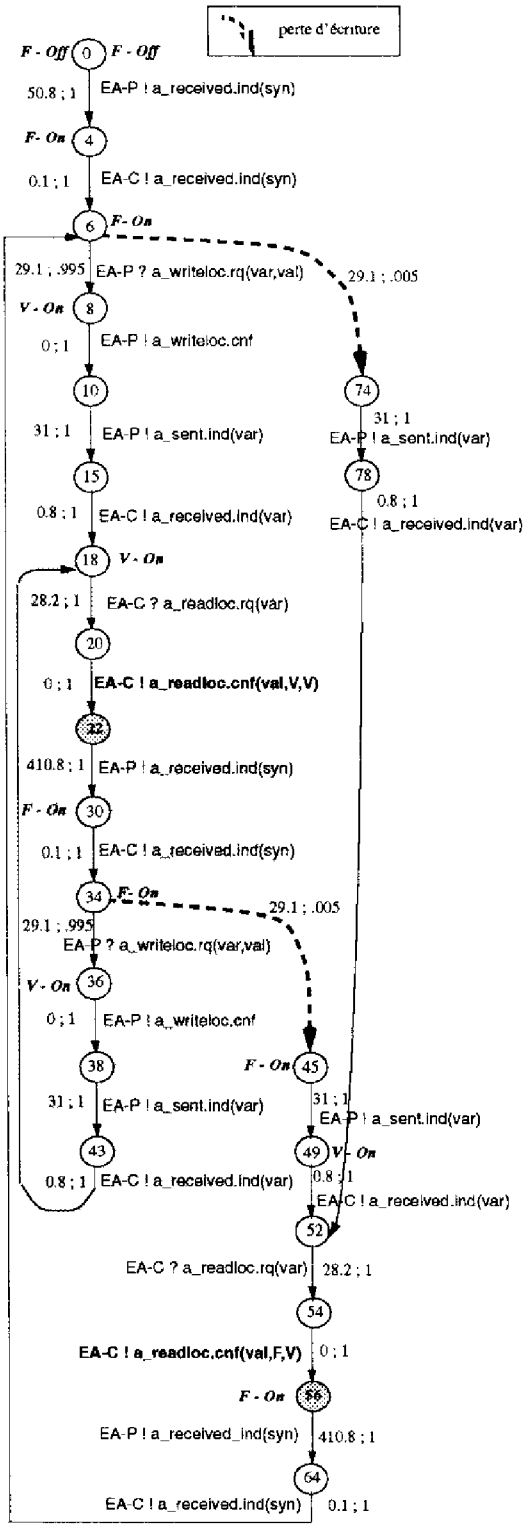

Figure 24: vue abstraite du service application (cas de pertes d'écriture)

* on a, avec une probabilité 0.005 et au bout d'un temps de $88.2\,Ut$, la confirmation de lecture au PU-C avec le status de rafraîchissement Faux (`EA-C!a_readloc.cnf(val,F,V)`) dans l'état 56 ;

– Évaluation du MTFF, suite à une requête d'écriture, d'occurrence d'une confirmation de lecture avec le status de rafraîchissement Faux :

$$MTFF(raf(F), pro(V)) = 100061\,Ut$$

4.4 Prise en compte d'une perte de synchronisation

La vue abstraite, représentée sur la figure 25, fait apparaître les propriétés suivantes :

- Analyse qualitative
  - on retrouve toujours le régime transitoire et le régime permanent du fonctionnement normal (partie grise) et on voit maintenant des chemins inhérents aux pertes de synchronisation ;
  - après `EA-P!a_received.ind(syn)` que ce soit en régime transitoire (état 4) ou en régime permanent (état 30), on a la potentialité d'une confirmation de lecture correcte (`EA-C!a_readloc.cnf(val,V,V)`) de l'état 20 et la potentialité d'une confirmation de lecture avec un status de promptitude Faux (`EA-C!a_readloc.cnf(val,V,F)`) dans l'état 63 ;
  - on remarque encore que suite à une confirmation de lecture incorrecte (état 63), on se resynchronise dans le régime permanent normal (état 71 → 34) ou on peut encore faire une confirmation de lecture avec le status de promptitude Faux, suite à l'occurrence d'une nouvelle perte de synchronisation (état 71 → 75... → 63).
- Analyse quantitative
  - Quantification des relations de potentialité précédentes : par exemple, dans l'état 30 du régime permanent normal, après `EA-P!a_received.ind(syn)` :
    * on a, avec une probabilité 0.995 et au bout d'un temps de $88.2\,Ut$ la confirmation de lecture (`EA-C!a_readloc.cnf(val,V,V)`) dans l'état 22 ;
    * on a, avec une probabilité 0.005 et aussi un temps de $88.2\,Ut$, la confirmation de lecture (`EA-C!a_readloc.cnf(val,V,F)`) dans l'état 63.
  - Évaluation du MTFF, suite à une requête d'écriture, d'occurrence d'une confirmation de lecture avec le status de promptitude Faux :

$$MTFF(raf(V), pro(F)) = 99559\,Ut$$

4.5 Prises en compte de pertes d'écriture et de synchronisation

La vue abstraite, représentée sur la figure 26, montre la combinaison des conséquences d'occurrences des deux types de pertes. On peut facilement faire des analyses qualitatives et quantitatives sur les différentes séquences.

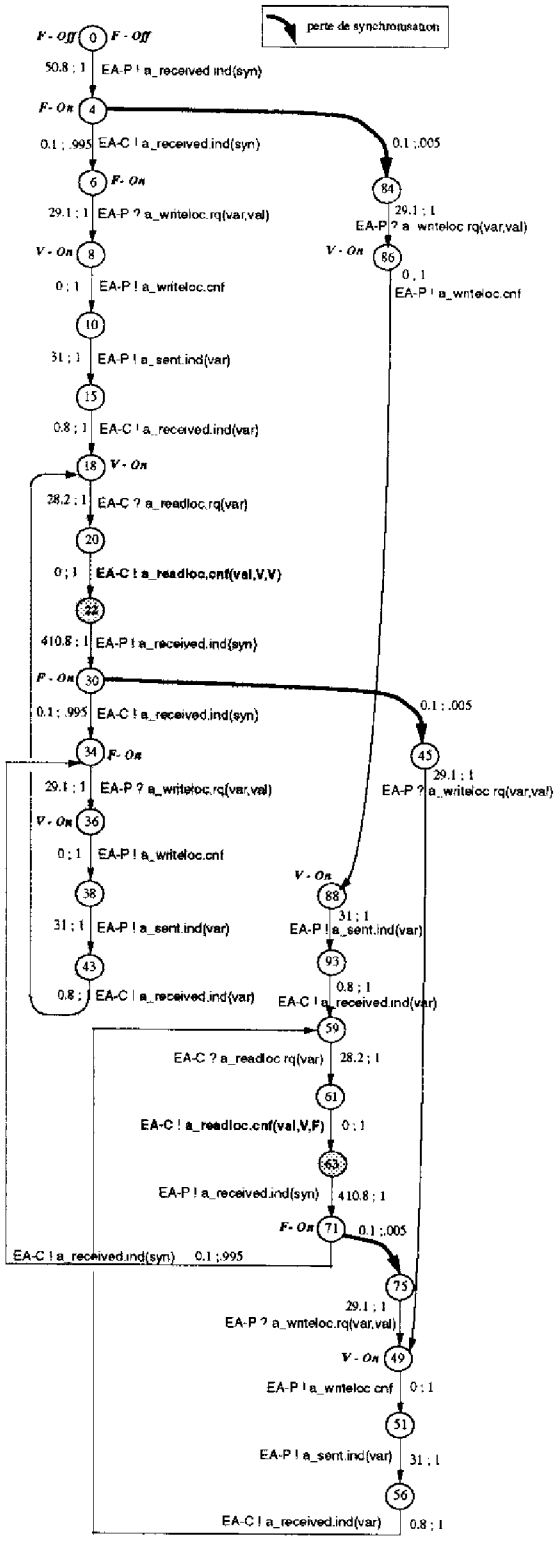

Figure 25: vue abstraite du service application (cas de pertes de synchronisation)

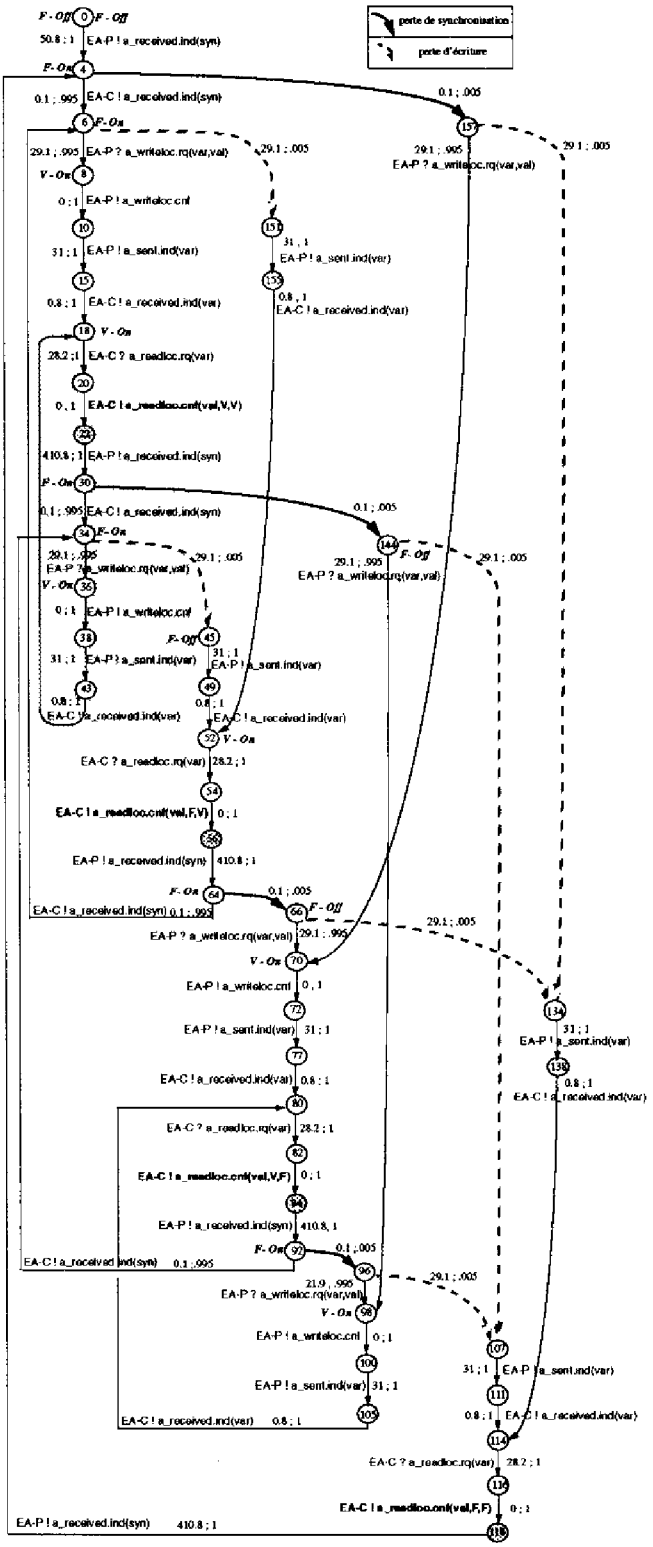

Figure 26: vue abstraite du service application (cas avec pertes d'écriture et de synchronisation)

Nous voulons plutôt insister ici sur l'aspect sûreté de fonctionnement de la relation producteur-consommateur, compte tenu des différentes combinaisons possibles pour les valeurs des status de rafraîchissement et de promptitude suite à une requête d'écriture. On obtient :

$$MTFF(raf(F), pro(V)) = 100562\,Ut$$
$$MTFF(raf(V), pro(F)) = 100060\,Ut$$
$$MTFF(raf(F), pro(F)) = 2.10^7\,Ut$$

5 Conclusion

L'étude effectuée dans ce chapitre constitue, à notre avis, une contribution importante au monde du réseau FIP en termes de représentation formelle et d'analyses permises.

La structuration proposée permet une représentation fine, précise et exhaustive des entités d'application et liaison à la fois en terme de modules internes et d'échanges internes (notions non spécifiées dans la norme et que nous avons définies) et de relations entre les échanges internes et les échanges externes (services spécifiés dans la norme). Ainsi, on a identifié :
- pour l'entité application producteur, les modules écriture locale, rafraîchissement, indication de réception de variable de synchronisation et indication d'émission de variable ;
- pour l'entité application consommateur, les modules lecture locale, promptitude, indication de réception de variable de synchronisation et indication de réception de variable.

Le modèle Réseaux de Petri Temporisés Stochastiques permet naturellement d'appréhender à la fois les niveaux de détail des modules internes et la composition de ces modules internes et des entités pour fournir un modèle global analysable.

L'analyse des mécanismes de validité temporelle (rafraîchissement et promptitude), associés aux services périodiques synchrones, a été effectuée tout d'abord, en supposant qu'il n'y a pas de pertes d'échanges et ensuite, en considérant deux hypothèses de pertes possibles (une perte d'écriture par le processus utilisateur producteur et une perte de trames de synchronisation sur le médium pour l'entité consommateur). Nous voulons souligner les résultats suivants obtenus :

- quand on n'a pas de pertes, on vérifie le bon fonctionnement cyclique des services application d'écriture locale et de lecture locale avec plus particulièrement, une confirmation de lecture avec les status de rafraîchissement et de promptitude vrais ;
- quand on a des hypothèses de pertes, on vérifie la capacité de FIP à détecter les conséquences de ces pertes sur les status dans la confirmation de lecture :
  - si on a une perte d'écriture, on a une confirmation de lecture avec le rafraîchissement faux ;
  - si on a une perte de synchronisation, on a une confirmation de lecture avec la promptitude fausse ;
  - si on a les deux types de pertes consécutivement, on a une confirmation de lecture avec les status de rafraîchissement et de promptitude faux...

L'analyse quantitative nous a permis en particulier, d'évaluer dans les différentes hypothèses de pertes, le temps moyen d'occurrence d'une erreur (MTFF) sur la confirmation de lecture. Cette étude, à la fois qualitative et quantitative, sur les mécanismes de rafraîchissement et de promptitude est, à notre avis, une contribution originale.

Chapitre 5

Contrôle-commande de poste de transformation d'énergie électrique réparti sur le réseau FIP : structuration, modélisation et analyse

1 Introduction

Ce chapitre présente l'application de la méthodologie, développée au chapitre 2, au système de contrôle-commande du poste de transformation d'énergie électrique de EDF réparti sur le réseau de terrain FIP. Plus précisément, nous considérons dans le cadre de cette étude une topologie de poste réduite à deux cellules départ et nous prenons comme hypothèse de défaut apparaissant sur les lignes départ, des défauts polyphasés qui sont immédiatement signalés aux cellules départ par le shunt (mais, nous ne représentons pas le shunt).
L'objectif de cette étude simplifiée est double :

étudier la propriété fonctionnelle importante de la sélectivité logique à savoir, l'évitement de défauts multiples ;
- analyser l'influence de la période de mise à jour des données sur le réseau FIP sur la mise en œuvre de la sélectivité logique.

Ce chapitre comprend trois parties :

- la première partie présente la structuration de l'application de contrôle-commande (la structuration du réseau FIP a déjà été présentée dans le chapitre 4) ;
- la seconde partie présente la modélisation du système étudié sur la base des Réseaux de Petri Temporisés Stochastiques ;
- la troisième partie présente l'analyse du fonctionnement de la sélectivité logique pour l'évitement des défauts multiples.

2 Structuration de l'application de contrôle-commande

L'architecture globale de l'application de contrôle-commande de poste EDF est directement issue de la figure 1 présentée au chapitre 2 et restreinte à deux cellules départ. La figure 1 représente les deux cellules départ D1 et D2 qui sont réparties sur le réseau de terrain FIP et qui ont pour rôle de contrôler et commander leur procédé respectif grâce : aux capteurs qui mesurent le courant électrique sur la ligne et aux disjoncteurs qui reçoivent les ordres de commande des cellules.

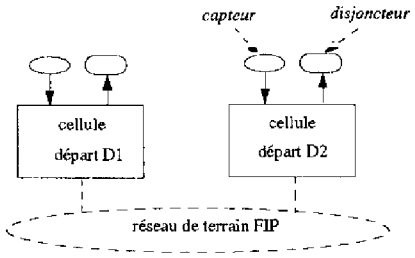

Figure 1: architecture globale réduite

Nous présentons ci-dessous l'architecture détaillée de l'application de contrôle-commande, c'est-à-dire d'une part, l'architecture d'une cellule départ et d'autre part, les différents échanges.

2.1 Cellule départ

Une cellule départ est décomposée selon les fonctions identifiées pour un sous-système de contrôle-commande local (cf. § 2.2 du chapitre 2).

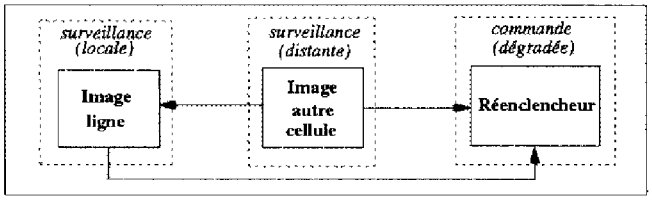

Figure 2: architecture détaillée d'une cellule départ

La figure 2 représente l'architecture détaillée d'une cellule départ qui comprend trois modules de plus haut niveau : le module *commande(dégradée)* (la commande normale n'a pas été représentée ici car nous étudions seulement la commande en cas de défaut électrique) et les modules *surveillance(locale)* et *surveillance(distante)*. Nous ne considérons pas ici le module *Interface réseau* (les interfaçages avec le réseau seront pris en compte directement par les modules que nous venons de définir).

Le module *surveillance(locale)* comprend le module élémentaire **Image ligne** (image des défauts électriques sur la ligne départ associée à la cellule).

Le module *surveillance(distante)* comprend le module élémentaire **Image autre cellule** (image des défauts sur la ligne départ associée à l'autre cellule).

Le module *commande(dégradée)* comprend le module élémentaire **Réenclencheur** (commande du disjoncteur en cas de défaillances).

Les flèches à l'intérieur de la cellule départ montrent les modules élémentaires en relation. Les modules *Image ligne* et *Image autre cellule* sont en relation pour détecter et diagnostiquer une défaillance à partir de l'état du procédé local et du procédé distant. Les modules *Image ligne*, *Image autre cellule* et *Réenclencheur* sont en relation pour décider des procédures de reprise à effectuer en cas de défaillances.

2.2 Échanges

Raisonnons sur la cellule départ D1 (le comportement de la cellule D2 étant identique). Les trois types d'échanges sont :

- tout d'abord, les échanges locaux entre la cellule départ D1 et son procédé local (capteur et disjoncteur) ;
- puis, les échanges distants entre la cellule départ D1 et l'autre cellule départ D2, qui sont en relation via le réseau de terrain FIP ;
- enfin, les échanges internes entre les modules élémentaires de la cellule départ D1.

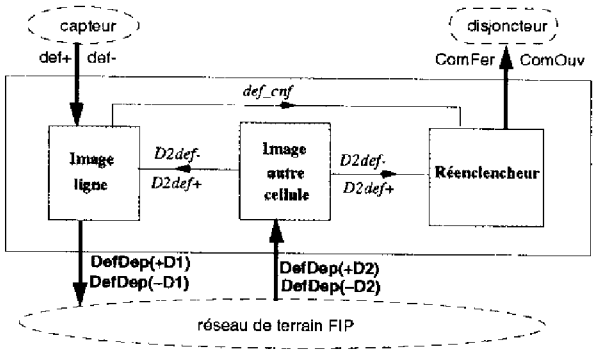

Figure 3: vue des échanges au niveau de la cellule départ D1

2.2.1 Échanges locaux

Les échanges entre une cellule départ et son environnement local constituent les échanges locaux relatifs au flux de contrôle et au flux de commande (cf. §2.2.3 du chapitre 2). Ils concernent (cf. figure 3) : les compte-rendus de l'état du procédé local en provenance des

capteurs (intelligents) pour signaler l'existence ou non d'un défaut électrique ; les ordres de commandes envoyés au disjoncteur (suite à la détection de défaillances).

- le module **Image ligne** gère :
  à l'interface avec le capteur, les messages relatifs à la détection de `défauts` électriques sur la ligne, c'est-à-dire : si un défaut a été détecté par le capteur (au bout du temps de préconfirmation), il est signalé à la cellule départ par le message `def+` ; si un défaut disparaît au niveau du capteur, il est signalé à la cellule par le message `def-` ;
- le module **Réenclencheur** gère :
  à l'interface avec le disjoncteur, les commandes envoyées au disjoncteur : le message `ComOuv` est envoyé pour `Commander` l'`Ouverture` du disjoncteur et le message `ComFer` est envoyé pour `Commander` sa `Fermeture`.

2.2.2 Échanges distants

Ces échanges concernent (cf. figure 3) les échanges à travers le réseau, c'est-à-dire les envois des messages signifiant que l'état courant de la ligne départ D1 est en défaut (**DefDep(+D1)**) ou n'est pas en défaut (**DefDep(-D1)**), ou les réceptions des messages signifiant que l'état courant de la ligne départ D2 est en défaut (**DefDep(+D2)**) ou n'est pas en défaut (**DefDep(-D2)**).

Le module **Image ligne** gère l'écriture des messages **DefDep(±D1)** dans l'interface de communication.

Le module **Image autre cellule** gère la lecture des messages **DefDep(±D2)** dans l'interface de communication.

2.2.3 Échanges internes

Les interactions entre les modules élémentaires constituent les échanges internes. Ils concernent (cf. figure 3) :
– l'échange def_cnf ("défaut confirmé") qui sert à signaler l'état en défaut de la ligne départ D1 et qui est émis par le module *Image ligne* vers le module *Réenclencheur* ;
– les échanges $D2def+$ (respectivement $D2def-$) qui servent à indiquer l'existence d'un défaut (respectivement l'absence de défaut) sur la cellule D2 distante et qui sont émis par le module *Image autre cellule* vers les modules *Image ligne* et *Réenclencheur*.

3 Modélisation

Nous donnons tout d'abord les modèles Réseaux de Petri des cellules départ, de leur environnement et du réseau FIP en représentant, comme au chapitre 4, les transitions immédiates

par des traits fins et les transitions temporisées par des traits gras. Ensuite, nous donnons les spécifications temporelles des transitions temporisées.

En ce qui concerne la présentation de ces modèles, nous procédons de la manière suivante :

- pour l'environnement et le réseau FIP, les modèles sont donnés directement avec les modules élémentaires composés ;
- pour la cellule départ, ce sont les modules élémentaires qui sont donnés séparément (la taille du modèle de la cellule départ étant trop grande).

Nous ne décrivons pas les différentes analyses que l'on peut faire par la construction de ces modèles (cf. chapitre 2 et 4).

Le modèle global, qui est obtenu par composition de ces modèles, n'est pas représenté ici.

3.1 Modèle de l'Environnement

Nous représentons l'Environnement de la cellule D1 (appelé E1) au moyen de trois modules élémentaires :

- un module, appelé **démon**, qui veut visualiser, de manière non déterministe sur un temps donné, un phénomène d'apparition ou de non apparition d'une perturbation électrique (qui a une durée) et qui va donc conditionner l'apparition ou la non apparition d'un défaut (pendant sa durée) ;
- un module, appelé **trait**, qui veut visualiser l'intelligence du capteur et qui représente le traitement effectué sur le signal électrique de la ligne départ durant la durée de préconfirmation ;
- un module, appelé **disj**, qui représente le comportement du disjoncteur départ.

Nous donnons directement sur la figure 4 le modèle de l'Environnement **E1** après composition des modules élémentaires en relation par fusion de places dupliquées et de transitions en rendez-vous.

Module *E1[démon]*
A partir de l'état initial **repos**, on a :
– soit, l'apparition de la perturbation (qui amène le module dans l'état **présent** par le tir de **t3**) au bout du temps **Td** (**Td** représente l'instant d'apparition de la perturbation électrique) ;
– soit, la non apparition (qui amène le module dans l'état **absent** par le tir de **t1**) au bout du temps **Td**.

Ensuite, à partir de ces états, le module passe dans l'état **fin** au bout du temps d'analyse considéré **Tf** (**Tf** représente l'instant de fin d'existence du défaut électrique, s'il était apparu) par le tir de la transition **t2** ou **t4**.

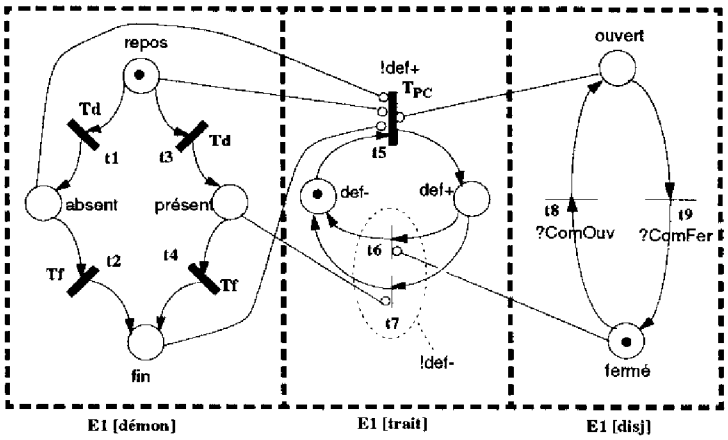

Figure 4: RdP du module E1

Notons qu'il est possible de concevoir une réinitialisation du démon, mais ceci représenterait une reproduction cyclique de l'apparition des défaillances (qui n'est pas considérée ici).

Module *E1[trait]*

Il représente le résultat du processus de traitement du signal électrique par le capteur intelligent. A l'initialisation, le processus n'a pas connaissance de l'existence d'un défaut ; il est dans l'état `def-`. Si à partir de cet état et pendant toute la durée de préconfirmation T_{PC}, le démon générant la perturbation est présent ($\neg$ *démon*[*repos*] $\wedge \neg$ *démon*[*absent*] $\wedge \neg$ *démon*[*fin*]) et le disjoncteur est fermé ($\neg$*disj*[*ouvert*]), alors la transition **t5** est tirée ; ceci génère l'émission du message d'existence de défaut à destination de la cellule départ (`!def+`) et amène le module dans l'état `def+`, indiquant ainsi qu'un défaut électrique existe. A partir de cet état, le module émet le message de disparition du défaut (`!def-`) à destination de la cellule départ et passe dans l'état `def-` lorsque le défaut disparaît parce que : soit, le disjoncteur a été ouvert ($\neg$*disj*[*fermé*]) (tir de **t6**) ; soit, la perturbation générant le défaut électrique a disparu ($\neg$*démon*[*présent*]) (tir de **t7**).

Module *E1[disj]*

Il représente l'état courant du disjoncteur. A partir de l'état initial `fermé`, la réception de la commande d'ouverture en provenance de la cellule (`?ComOuv`) amène le disjoncteur dans l'état `ouvert` ; à partir de cet état, la réception de la commande de fermeture de la cellule départ (`?ComFer`) le ramène dans l'état `fermé`.

3.2 Modèle d'une cellule départ

La modélisation d'une cellule départ présentée ici ne concerne qu'une partie des automatismes de poste complets décrits au moyen des Réseaux de Petri Temporisés Stochastiques dans [BS95].

Le modèle que nous considérons (qui est suffisant dans le contexte de l'étude réalisée dans

ce chapitre) comprend trois modules élémentaires (cf. figure 2) : **Image ligne**, appelé **lig**, **Image autre cellule**, appelé **aut**, **Réenclencheur**, appelé **rll**. Ces modèles sont représentés respectivement sur les figures 5, 6 et 7.

Module *D1[lig]* (figure 5)
A l'initialisation, le module est dans l'état `no_def` ("sans défaut"). Lorsqu'un défaut est signalé, par le capteur "intelligent", à la cellule par la réception du message de présence de défaut (`?def+`), le module passe dans l'état `def_cnf` : soit, par le tir de **t1** parce que l'autre départ D2 est en défaut ($\neg aut(D2def-)$) (il faut confirmer immédiatement le défaut local afin d'éviter/diminuer l'existence du défaut multiple par l'ouverture immédiate du départ D1) ; soit, par le tir de **t2** parce que l'autre départ D2 n'est pas en défaut ($\neg aut(D2def+)$). Le module passe dans l'état `def_cnf` en activant le délai de surveillance de disparition du défaut `tmp_sur_def` (de durée T_{sur}) ; ce délai permet de vérifier que le défaut disparaît bien en un temps limité après sa confirmation.

A partir de l'état `def_cnf`, dès que le message de disparition du défaut est reçu du capteur "intelligent" (`?def-`), le module revient à l'état `no_def` et le délai `tmp_sur_def` est désactivé (tir de **t4**). La fin du délai de surveillance est représentée par la transition **t3**, si le défaut ne disparaît pas avant la fin de T_{sur}.

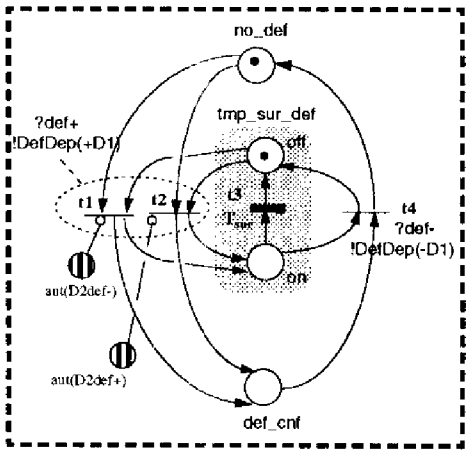

Figure 5: RdP du module D1[lig]

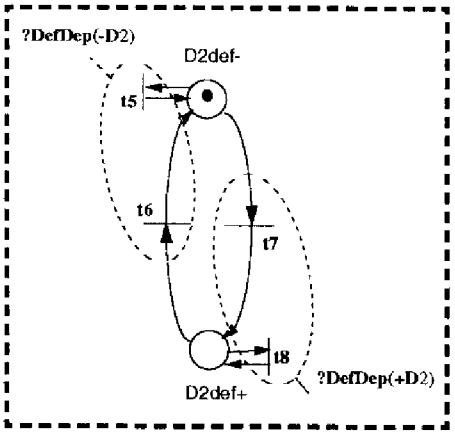

Figure 6: RdP du module D1[aut]

Module *D1[aut]* (figure 6)
L'image des autres cellules est restreinte à l'état de la cellule distante D2 qui est, à l'initialisation, dans l'état `D2def-` ("pas de défaut sur D2").
La réception de message d'absence de défaut sur D2 (**?DefDep(-D2)**), via le réseau, provoque :
- si le module est dans l'état initial, il y reste (tir de **t5**) ;
- si le module est dans l'état `D2def+`, il retourne dans l'état initial (tir de **t6**).

Si un message d'existence de défaut sur le départ D2 est reçu (**?DefDep(+D2)**), alors :
- si le module est dans l'état initial, il passe dans l'état `D2def+` (tir de **t7**) ;

– si le module est dans cet état, il y reste (tir de **t8**).

Modules *D1[rll]* (figure 7)

Nous représentons uniquement les éléments de modélisation nécessaires à la réalisation de l'étude pendant la durée des perturbations considérées, c'est-à-dire : le cycle rapide et une partie du cycle lent_1. Des délais, appelés délais de verrouillage, sont associés aux cycles de réenclenchement, afin de garantir que chaque cycle ne sera pas exécuté plusieurs fois consécutivement. Ces délais sont activés dès le début de la fermeture automatique de leur cycle respectif et sont appelés respectivement `tmp_verr_rap` (de durée TV_{rap}) et `tmp_verr_lt1` (de durée TV_{lt1}). On dit qu'un cycle est verrouillé si son délai de verrouillage est actif. Un cycle verrouillé ne peut plus être effectué.

A l'initialisation, le module D1[rll] est dans l'état `repos`. Dès qu'un défaut est confirmé dans le module D1[lig] ($\neg lig(no\_def)$), ce module envoie la commande d'ouverture au disjoncteur (`!ComOuv`) pour exécuter :
– l'ouverture rapide dans la place `ouv_rap` (tir de **t11**) si les cycles rapide et lent_1 ne sont pas verrouillés ($\neg tmp\_verr\_rap(on) \wedge \neg tmp\_verr\_lt\_1(on)$) et le délai d'ouverture rapide est alors activé ;
– l'ouverture lent_1 dans la place `ouv_lt1` (tir de **t21**) si le cycle rapide est verrouillé et pas le cycle lent_1 ($\neg tmp\_verr\_rap(off) \wedge \neg tmp\_verr\_lt\_1(on)$) et le délai d'ouverture lent_1 est alors activé.

Lorsque le délai d'ouverture rapide `tmp_ouv_rap` (de durée TO_{rap}) arrive à terme, le délai d'attente de disparition de défauts des autres cellules `tmp_wait_out` (de durée T_{wait}) est simultanément activé (tir de **t12**). Si aucun défaut n'est signalé par la cellule D2 ($\neg aut(D2def+)$), cette temporisation est immédiatement désactivée (tir de **t14**) ; sinon, la temporisation reste activée tant qu'un défaut est signalé par la cellule départ D2 ($\neg aut(D2def-)$). Au bout de la durée du délai T_{wait} (tir de **t13**), si le défaut sur D2 n'a pas disparu, le disjoncteur reste ouvert (en attendant une commande de l'opérateur, non considéré ici).

A partir de l'état `ouv_rap`, la commande de fermeture `!ComFer` est envoyée au disjoncteur (tir de **t16**) : si les temporisations `tmp_ouv_rap` et `tmp_wait_aut` ne sont pas actives et s'il n'existe pas de défaut connu sur la cellule départ D2 ($\neg aut(D2def+)$). Le délai de verrouillage rapide `tmp_verr_rap` et le délai de fermeture `tmp_fer` (de durée TF) sont alors activés.
Le délai de fermeture TF arrive à terme au bout de **TF** (tir de **t19**).
Si le délai de fermeture n'est plus actif ($\neg tmp\_fer(on)$), le module retourne normalement dans l'état `repos` (tir de **t15**).

A partir de l'état `fer`, si un défaut multiple est identifié par la connaissance d'un défaut local confirmé ($\neg lig(no\_def)$) et d'un défaut sur la cellule D2 ($\neg aut(D2def-)$), alors que la temporisation de fermeture n'est pas arrivée à son terme, la commande d'ouverture est immédiatement envoyée au disjoncteur départ local (`!ComOuv`) pour effectuer l'ouverture en

cycle lent_1 car le cycle rapide est verrouillé ($\neg tmp\_verr\_rap(off)$) et pas le cycle lent_1 ($\neg tmp\_verr\_lt1(on)$) (tir de **t20**). Le module passe alors dans l'état `ouv_lt1` et la temporisation de fermeture est désactivée, tandis que la temporisation tmp_verr_lt1 est activée.

Lorsque le délai d'ouverture lent_1 tmp_ouv_lt1 (de durée TO_{lt1}) arrive à sa fin, le délai d'attente de disparition de défaut des autres cellules tmp_wait_aut est simultanément activé (tir de **t22**). A partir de l'état `ouv_lt1`, et de façon identique que pour la fermeture à partir de l'état `ouv_rap`, à la fin de la temporisation d'ouverture lent_1, si les temporisations tmp_ouv_lt1 et tmp_wait_out ne sont pas actives et s'il n'existe pas de défaut connu sur la cellule D2 ($\neg aut(D2def+)$), la commande de fermeture amène le module dans l'état `fer`. Le délai de verrouillage du cycle lent_1 est activé, ainsi que la temporisation de fermeture `tmp_fer` (tir de **t23**).

3.3 Modèle du réseau FIP

3.3.1 Choix de simplification

Comme nous voulons seulement analyser la *propriété d'évitement des défauts multiples* en fonction de différentes périodes de mise à jour réseau des données relatives aux états des cellules D1 et D2, nous considérons, comme modèle du réseau de terrain FIP, un modèle simplifié réduit au strict minimum pour les échanges de données à contraintes temporelles fortes utilisant les services périodiques de transfert de buffers (la couche application n'est pas nécessaire).

En ce qui concerne la couche liaison, nous adoptons une vue abstraite qui ne représente pas explicitement les buffers de stockage (les valeurs des variables à transmettre sont directement lues par des accès en lecture dans les modules *Image ligne* de chaque cellule), mais nous mettons l'accent sur les trois modules élémentaires essentiels pour une mise à jour réseau :

- le module qui représente la génération périodique, par l'arbitre de bus, des trames **id_dat** qui permettent d'effectuer la mise à jour (nous appelons ce module `Per`). Les trames **id_dat**, destinées aux cellules D1 et D2, sont appelées respectivement **id_dat(D1)** et **id_dat(D2)** où **D1** et **D2** sont les identifieurs associés ;
- le module qui représente les mécanismes pour la mise à jour sur D2 de l'état de D1 (nous appelons ce module `D1_vers_D2`). Il réalise : tout d'abord, la transmission de la trame **id_dat(D1)** de l'arbitre de bus vers la cellule D1, puis la transmission de la trame **rp_dat** contenant la variable indiquant l'état de défaut (+) ou non (-) de D1 (**rp_dat(Def_Dep(±D1))**) de la cellule D1 vers la cellule D2 ;
- le module qui représente les mécanismes pour la mise à jour sur D1 de l'état de D2 (nous appelons ce module `D2_vers_D1`). Les mécanismes représentés sont totalement identiques à ceux du point ci-dessus en remplaçant D1 par D2 et vice versa.

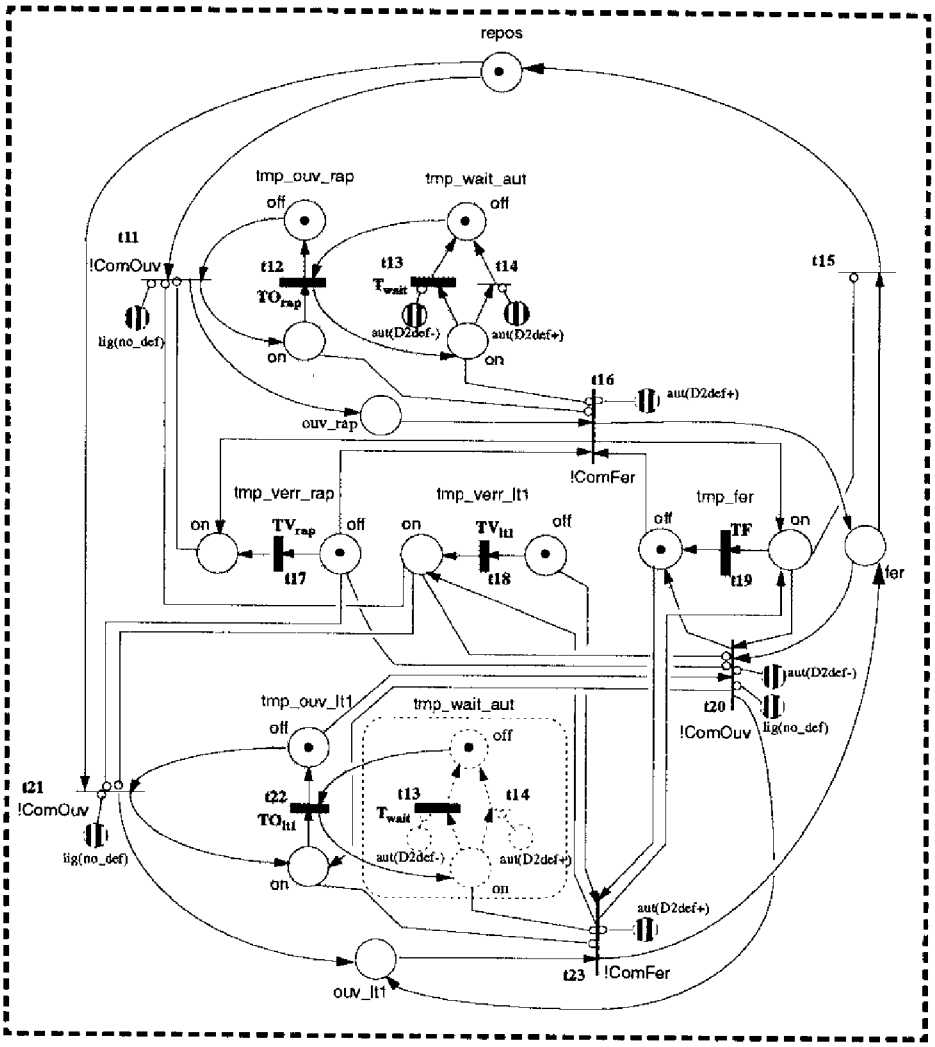

Figure 7: RdP du module D1[rll]

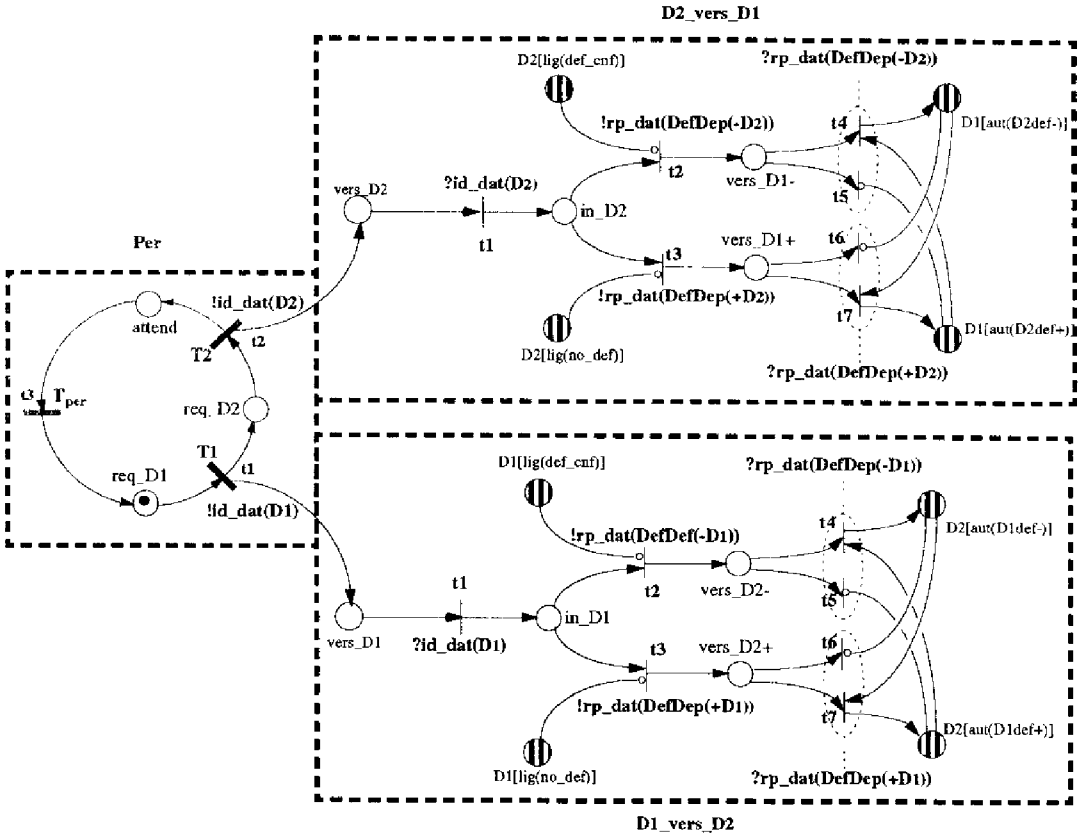

Figure 8: modèle simplifié du réseau FIP

Compte tenu des ordres de grandeur des durées de transmission des trames sur le médium par rapport aux constantes de temps de l'application, les délais de transmission sur le médium sont négligés (ils sont donc représentés ici par des transitions immédiates).

3.3.2 Le modèle considéré

Ce modèle est donné sur la figure 8 où nous avons représenté directement la composition des modules `Per`, `D1_vers_D2` et `D2_vers_D1`.

Module *Per*

A l'initialisation, le module est dans l'état `req_D1`. Au bout du temps $T1$, la trame requête de D1 est émise sur le médium (**!id_dat(D1)**) (tir de **t1**) et le module passe dans l'état `req_D2`. Au bout du temps $T2$, la trame requête de D2 est émise sur le médium (**!id_dat(D2)**) (tir de **t2**) et le module passe dans l'état **attend**. Dans cet état, il attend la fin du temps T_{per} (tir de **t3**) qui correspond à la durée d'attente pour réaliser la période du micro-cycle ($T_{\mu C} = T1 + T2 + T_{per}$).

Module *D1_vers_D2*

Lorsque la trame requête **id_dat(D1)** est reçue dans la place **vers_D1**, elle est transmise jusqu'à la cellule D1 (**?id_dat(D1)**) (tir de **t1**) et le module passe dans la place **in_D1** en attente de la réponse de la cellule D1 :

- soit avec la valeur (-) ($\neg D1[lig(def\_cnf)]$) indiquant que la cellule D1 n'est pas en défaut (tir de **t2**) en émettant la trame réponse **!rp_dat(DefDep(-D1))**. Le module passe dans la place **vers_D2-**. Cette trame est alors reçue et stockée dans le module *Image autre cellule* de la cellule D2 :
  - si l'ancienne valeur stockée est différente, le tir de la transition **t4** amène le jeton dans la place **D2[aut(D1def-)]** ;
  - si l'ancienne valeur stockée est identique ($\neg D2[aut(D1def+)]$), le tir de la transition **t5** ne met pas à jour la place **D2[aut(D1def-)]** ;

- soit avec la valeur (+) ($\neg D1[lig(no\_def]$) indiquant que la cellule D2 est en défaut (tir de **t3**) en émettant la trame réponse **!rp_dat(DefDep(+D1))**. Le module passe dans la place **vers_D2+**. Cette trame est alors reçue et stockée dans le module *Image autre cellule* de la cellule D2 :
  - si l'ancienne valeur stockée est différente, le tir de la transition **t7** amène le jeton dans la place **D2[aut(D1def+)]** ;
  - si l'ancienne valeur stockée est identique ($\neg D2[aut(D1def-)]$), le tir de la transition **t6** ne met pas à jour la place **D2[aut(D1def+)]**.

Module *D2_vers_D1*

Ce module est symétrique au module **D1_vers_D2** en remplaçant D1 par D2 et vice versa.

3.4 Spécifications temporelles

Les caractéristiques temporelles qui sont nécessaires à l'étude sont :

- les spécifications temporelles des perturbations électriques générées sur les lignes départ D1 et D2 (représentées dans le module **démon** de l'environnement).

  Nous considérons un chevauchement de ces perturbations de manière à créer la potentialité d'un défaut départ multiple. Les scénarios d'apparition et de disparition des défauts sont représentés sur la figure 9 où la perturbation sur D1, respectivement D2, existe sur l'intervalle [40 ms ; 740 ms], respectivement [180 ms ; 730 ms].

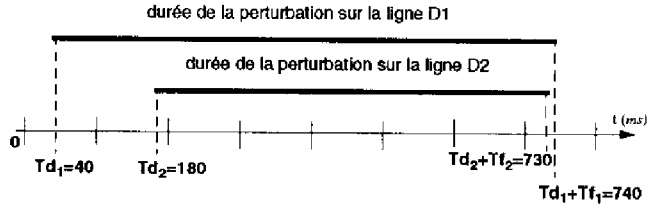

Figure 9: caractéristiques temporelles des perturbations électriques

Module	Transition	Distribution temporelle $f(x)$
E1[démon]	t1, t3	$\delta(x-400)$
	t2, t4	$\delta(x-7000)$
E2[démon]	t1, t3	$\delta(x-1500)$
	t2, t4	$\delta(x-5800)$
Ei[trait] (i=1,2)	t5	$\delta(x-1000)$
Di[lig] (i=1,2)	t3	$\delta(x-20000)$
Di[rll] (i=1,2)	t12	$\delta(x-3000)$
	t13	$\delta(x-20000)$
	t17	$\delta(x-30000)$
	t18	$\delta(x-100000)$
	t19	$\delta(x-5000)$
	t22	$\delta(x-150000)$
Per (r=1)	t1,t2	$\delta(x-100)$
	t3	$\delta(x-800)$
Per (r=2)	t1,t2	$\delta(x-100)$
	t3	$\delta(x-300)$

Tableau 1: spécifications temporelles

- les spécifications temporelles des différentes temporisations associées aux mécanismes de la sélectivité logique (représentées dans le module réenclencheur `rll`). Compte tenu des caractéristiques temporelles des perturbations électriques (qui activent les cycles de réenclenchement rapide pour les cellules D1 et D2 et le début du cycle de réenclenchement lent_1 pour la cellule D2), nous spécifions les valeurs des temporisations suivantes : $\mathbf{TO}_{rap}$=300ms ; **TF**=500ms ; $\mathbf{TO}_{lt1}$=15s ; $\mathbf{TV}_{rap}$=3s et $\mathbf{TV}_{lt1}$=10s ;

- les spécifications temporelles des caractéristiques de l'arbitre de bus du réseau FIP (module `Per`), c'est-à-dire de mise à jour (ou durée du micro-cycle $\mathbf{T}_{\mu C}$) des variables **DefDep(±D1)** et **DefDep(±D2)** et d'autre part, les instants d'envoi des trames **id_dat(D1)** et **id_dat(D2)**. Concernant la périodicité de mise à jour, nous choisissons deux périodes obtenues pour les valeurs 1 et 2 de r (cf. équation 3.1 du chapitre 3) : si $r = 1$, $T_{\mu C} = 100\,ms$ et si $r = 2$, $T_{\mu C} = 50\,ms$. Les instants initiaux d'envoi des trames **id_dat(D1)** et **id_dat(D2)** correspondent respectivement aux dates $10\,ms$ et $20\,ms$. Le premier cas ($r = 1$) correspond au cas où la contrainte temporelle n'est pas respectée ; le second cas ($r = 2$) correspond au cas où la contrainte temporelle est respectée.

En adoptant, comme au chapitre 4, l'Unité de temps Ut égale à $0.1\,ms$, les spécifications temporelles des transitions temporisées des différents modèles sont celles données dans le tableau 1. Les transitions non citées sont des transitions immédiates (donc de distributions $f(x) = \delta(x)$).

4 Analyse

4.1 Cadre de l'analyse

Afin d'analyser uniquement le comportement du système en présence des perturbations électriques, nous considérons le modèle du démon réduit aux transitions **t3** et **t4** (la perturbation se produit donc de manière certaine) et nous limitons la durée d'analyse à une valeur un peu supérieure à la durée des perturbations électriques (de manière à vérifier le retour à la normale).

Nous considérons deux cas d'analyse : le cas 1 où la période de mise à jour est de $100\,ms$; le cas 2 où la période de mise à jour est de $50\,ms$. Ces deux cas donnent des graphes d'états probabilisés acycliques (conséquence de la limitation de la durée d'analyse) qui ont respectivement 121 états (cas 1) et 183 états (cas 2). Nous voulons ici focaliser sur le mécanisme de la sélectivité logique. Nous considérons donc une vue abstraite du graphe d'états probabilisé (les numéros des états donnés représentés sont ceux du graphe d'états probabilisé complet). Les vues abstraites sont obtenues en faisant une projection sur les états atteints après une interaction entre les cellules départ et le procédé (`def+`, `def-`, `ComOuv`, `ComFer`). Les arcs de la vue abstraite sont étiquetés avec : à droite, les échanges avec le procédé et à gauche, les valeurs du couple (*date_tir,probabilité_tir*) relatives au changement d'état.

Nous effectuons maintenant l'analyse de la sélectivité logique dans le cas 1, puis le cas 2 en présentant tout d'abord, les diagrammes des scénarios temporels et ensuite, les vues abstraites obtenues.

4.2 Analyse de la sélectivité logique

4.2.1 Diagrammes temporels des scénarios analysés

Des diagrammes temporels, dont le but est d'expliquer les scénarios du cas 1 et du cas 2, sont représentés respectivement sur les figures 10 et 11. Ces figures comprennent trois parties : les parties du haut et du bas représentent l'évolution temporelle des éléments associés aux comportements respectifs des cellules D1 et D2 (ces éléments sont définis au point A ci-dessous). La partie du milieu visualise les instants de réception, par la cellule D2, des trames **rp_dat(DefDep(±D1))** envoyées par la cellule D1 (la méthode de visualisation est indiqué au point B ci-dessous).

Notons que, comme la perturbation électrique sur la ligne D1 précède la perturbation électrique sur la ligne D2, la problématique est la réception des trames **rp_dat(DefDep(±D1))** par la cellule D2 en temps voulu (afin de faire jouer les mécanismes de la sélectivité logique). Par contre, la réception des trames **rp_dat(DefDef(±D2))** par la cellule D1 n'est pas ici pertinente et n'a donc pas été représentée.

A / Les scénarios présentent pour chaque cellule départ, D1 et D2 :
- la durée de la perturbation électrique (appelée démon E1 et démon E2 respectivement sur les lignes D1 et D2),
- les variations de l'intensité électrique sur la ligne départ concernée,
- la connaissance que la cellule a de l'état de l'autre cellule : autre cellule Dj en défaut, noté `aut(Djdef+)` ou autre cellule pas en défaut, noté `aut(Djdef-)` pour $j = (1, 2)$.
- la connaissance que la cellule a de l'état de sa ligne départ : ligne en défaut, noté `lig(def_cnf)` ou ligne pas en défaut, noté `lig(no_def)` ;

B / La réception périodique, au niveau de la cellule D2, des trames réponses **rp_dat(DefDep(±D1))** (contenant l'état de la cellule D1) est représentée par des traits verticaux en pointillé entre l'état `D1[lig(no_def ou def_cnf)]` et l'état `D2[aut(D1def±)]`. La variable **DefDep(±D1)** est transmise avec la valeur : − si au niveau de la cellule D1, le trait gras est sur la ligne `lig(no_def)` ; + si au niveau de la cellule D1, le trait gras est sur la ligne `lig(def_cnf)`.
L'instant de la première réception est fixé à la date $100\,Ut$. Comme la période du micro-cycle est de $1000\,Ut$ dans le cas 1 et de $500\,Ut$ dans le cas 2, les instants de réception des trames **rp_dat(DefDep(±D1))** sont donc : 100, 1100, 2100, ... pour le cas 1; 100, 600, 1100, 1600, ... pour le cas 2.
Au niveau de la cellule D2, la valeur de la variable **DefDep(±D1)** reçue est stockée et sa valeur est représentée par le trait en gras sur la ligne `aut(D1def-)` si elle est négative et sur la ligne `aut(D1def+)` si elle est positive.

4.2.1.1 Cas 1 ($T_{\mu C} = 1000\,Ut$)

Observons (figure 10), tout d'abord, l'évolution de la cellule départ D1 où le défaut apparaît à la date $400\,Ut$ et est préconfirmé à l'instant $1400\,Ut$ (le temps de préconfirmation $\mathbf{T}_{PC}$ ($1000\,Ut$) étant nécessaire au capteur intelligent pour analyser le signal). A cet instant $1400\,Ut$, la cellule reçoit l'information de défaut (`?def+`) et commande donc immédiatement l'ouverture en cycle rapide (`!ComOuv`), ce qui va provoquer la disparition immédiate du défaut (et donc la réception de l'information "pas de défaut" (`?def-`)). C'est ce qui explique la pointe représentée à l'instant $1400\,Ut$ sur le diagramme relatif à l'état de la ligne (`lig(no_def)`).

Au bout du délai $\mathbf{TO}_{rap}$ ($3000\,Ut$), à la date $4400\,Ut$, comme la cellule ne connaît pas l'existence de défaut sur la ligne D2 (`aut(D2def-)`), elle ordonne la fermeture (`!ComFer`) du disjoncteur pour une durée **TF** ($5000\,Ut$). Compte tenu de la durée d'existence du démon E1, un défaut est à nouveau signalé par la protection à la cellule à la date $5400\,Ut$ (`?def+`). Le défaut local est représenté sur la ligne `lig(def_cnf)`. Lorsque le défaut local disparaît (`?def-`) à la date $7400\,Ut$, ceci est représenté sur la ligne `lig(no_def)`. Ainsi, à la fin du délai de fermeture TF, compte tenu qu'il n'existe plus de défaut local, le disjoncteur est laissé fermé.

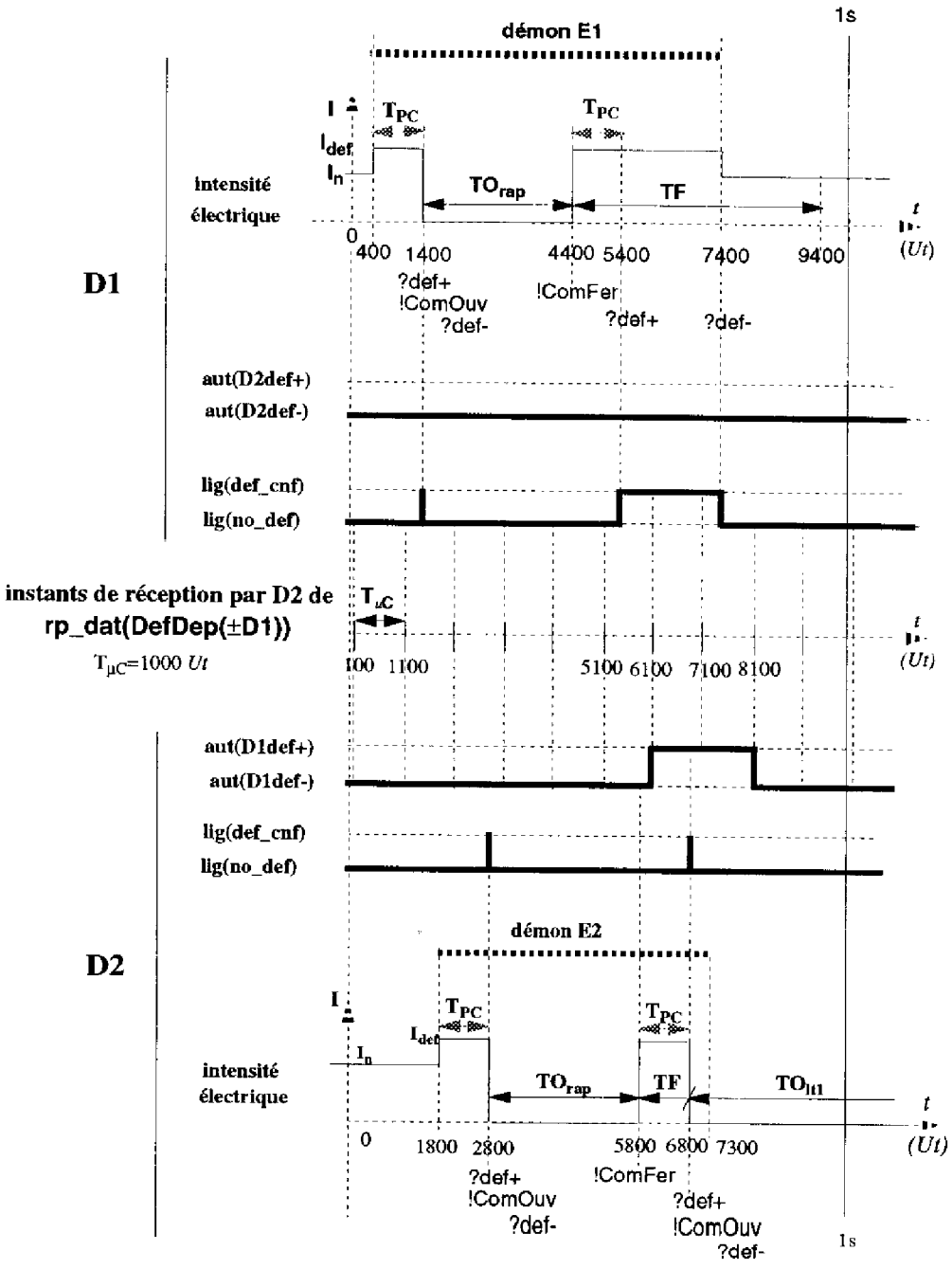

Figure 10: diagramme temporel relatif au cas 1

Jusqu'à l'instant 5100 Ut, la cellule D1 envoie des trames **rp_dat(DefDep(−D1))**. Elle envoie par contre des trames **rp_dat(DefDep(+D1))** aux instants 6100 Ut et 7100 Ut. Notons que le défaut local est à nouveau signalé lorsque le disjoncteur est fermé, à l'instant 5400 Ut mais, cette confirmation n'est transmise à la cellule D2, au plus tôt, qu'à l'instant 6100 Ut, c'est-à-dire avec un retard de 700 Ut. Ceci est très important comme nous le voyons maintenant en expliquant l'évolution de la cellule D2.

En ce qui concerne la cellule D2, le défaut local (qui apparaît à l'instant 1800 Ut) est préconfirmé à l'instant 2800 Ut, ce qui entraîne, à cet instant là, des actions similaires à celles présentées sur le diagramme de la cellule D1 (à l'instant 1400 Ut) : `?def+`, `!ComOuv`, `?def-`. O a donc l'existence d'une pointe sur l'état de la ligne `lig(def_cnf)` au démarrage de la temporisation $\mathbf{TO}_{rap}$ du cycle d'ouverture rapide. Au bout du délai $\mathbf{TO}_{rap}$, à la date 5800 Ut, comme la cellule D2 ne connaît pas l'existence de défaut sur la ligne D1 (`aut(D2def-)`), elle ordonne la fermeture du disjoncteur (`!ComFer`). Or un défaut préconfirmé existe déjà sur la ligne D1 (depuis l'instant 5400 Ut) mais, il ne sera connu qu'à l'instant 6100 Ut (du fait du retard indiqué précédemment). Donc ce n'est que lorsque la cellule D2 est informée de l'existence du défaut local à l'instant 6800 Ut (`?def+`) qu'elle entreprend les actions de commande d'ouverture de son disjoncteur (`!ComOuv`) c'est-à-dire, elle passe dans le cycle lent 1.

On voit donc que du fait du retard dans la connaissance que la cellule D2 a de la ligne D1, un défaut double existe entre les dates 5800 Ut **et** 6800 Ut.

4.2.1.2 Cas 2 ($T_{\mu C} = 500\,Ut$)

Nous commentons simplement (figure 11) les différences par rapport au cas 1 qui montre qu'une période de mise à jour plus petite (500 Ut) permet d'éviter l'existence du défaut double. Le diagramme temporel de la cellule D1 et le début du diagramme temporel de la cellule D2 sont identiques au cas précédent.

Par contre, on voit maintenant que, concernant l'état de défaut qui apparaît sur la ligne D1, à l'instant 5400 Ut, la cellule D2 en est informée à l'instant 5600 Ut et donc, à la fin du délai $\mathbf{TO}_{rap}$ (instant 5800 Ut), la cellule D2 prolonge alors l'ouverture de son disjoncteur ($\mathbf{T}_{wait}$) jusqu'à ce qu'elle soit informée de la disparition du défaut sur la ligne D1 (instant 7600 Ut).

On voit donc que dans ce cas, le défaut multiple est évité grâce à la prolongation de la phase d'ouverture rapide de 5800 Ut **à** 7600 Ut.

4.2.2 Les vues abstraites

4.2.2.1 Cas 1

La vue abstraite est représentée sur la figure 12. On a seulement une séquence d'évènements (ce qui est logique puisqu'on considère l'occurrence certaine des perturbations) ; ce qui induit d'une part, la propriété qualitative d'inévitabilité et d'autre part, des probabilités de

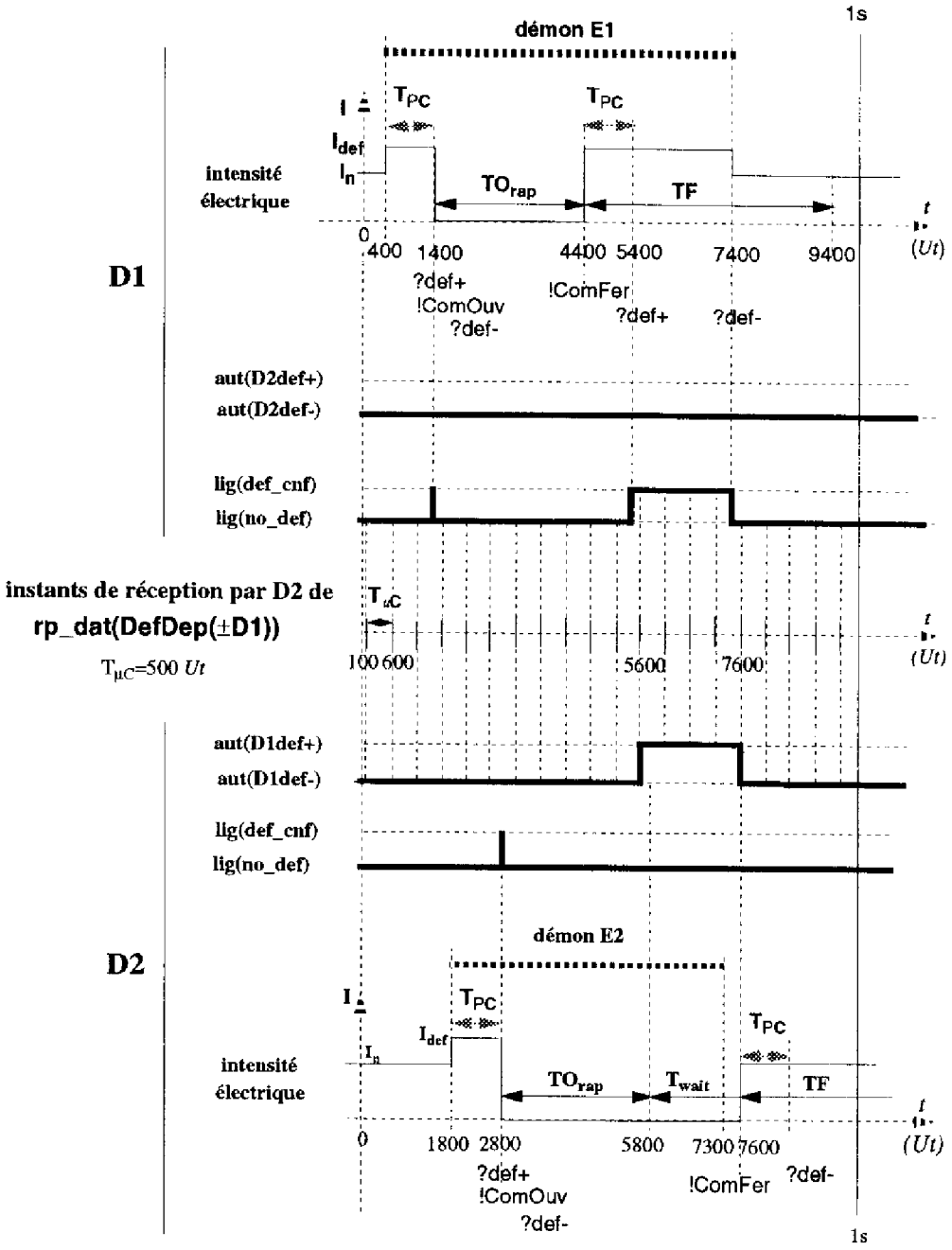

Figure 11: diagramme temporel relatif au cas 2

branchement égales à 1. L'étude concerne donc l'enchaînement des évènements.

- Analyse qualitative :
  - les échanges entre la cellule départ D1 et son environnement montrent que, après l'apparition du premier défaut ($\stackrel{\texttt{D1?def+}}{\longrightarrow}$ 12), la commande d'ouverture est envoyée au disjoncteur ($\stackrel{\texttt{D1!ComOuv}}{\longrightarrow}$ 13) ; cette ouverture provoque inévitablement la disparition du défaut ($\stackrel{\texttt{D1?def-}}{\longrightarrow}$ 14) ;
  - de même au niveau du départ D2, la première apparition du défaut ($\stackrel{\texttt{D2?def+}}{\longrightarrow}$ 27) provoque une commande d'ouverture du disjoncteur ($\stackrel{\texttt{D2!ComOuv}}{\longrightarrow}$ 28) ; cette ouverture amène inévitablement à la disparition du défaut ($\stackrel{\texttt{D2?def-}}{\longrightarrow}$ 30) ;
  - à la fermeture automatique du départ D1 ($\stackrel{\texttt{D1!ComFer}}{\longrightarrow}$ 41), le défaut est de nouveau signalé à la cellule ($\stackrel{\texttt{D1?def+}}{\longrightarrow}$ 52), ce qui signifie que le démon générant la perturbation électrique est toujours présent ;
  - puis, la cellule D2 commande également la fermeture automatique de son disjoncteur (suite à son état d'ouverture précédent 28) ($\stackrel{\texttt{D2!ComFer}}{\longrightarrow}$ 56). Le défaut est de nouveau signalé ($\stackrel{\texttt{D2?def+}}{\longrightarrow}$ 67) (i.e. le démon générant la perturbation électrique est toujours présent), ce qui provoque instantanément l'envoi de la commande d'ouverture du disjoncteur et donc la disparition du défaut sur la ligne D2 ($\stackrel{\texttt{D2!ComOuv}}{\longrightarrow}$ 68 $\stackrel{\texttt{D2?def-}}{\longrightarrow}$ 70) ;
  - ensuite, le défaut départ D1 disparaît ($\stackrel{\texttt{D1?def-}}{\longrightarrow}$ 74) car la perturbation électrique a cessé.

  On constate donc que l'on a l'existence d'un défaut multiple dans l'état 67 (défaut D1 noté depuis l'état 52 et défaut D2 noté dans l'état 67).

- Analyse quantitative :
  - le défaut est signalé à la cellule D1 à la date $1400\,Ut$ avec la probabilité égale à 1 et la commande d'ouverture en cycle rapide est alors immédiatement envoyée, puis le défaut disparaît immédiatement ($\stackrel{1400;1}{\longrightarrow}$ 12 $\stackrel{0;1}{\longrightarrow}$ 13 $\stackrel{0;1}{\longrightarrow}$ 14) ;
  - le défaut est signalé à la cellule D2 à la date $2800\,Ut$ avec la probabilité égale à 1 et la commande d'ouverture en cycle rapide est alors immédiatement envoyée, puis le défaut disparaît immédiatement ($\stackrel{1400;1}{\longrightarrow}$ 14 $\stackrel{1400;1}{\longrightarrow}$ 27 $\stackrel{0;1}{\longrightarrow}$ 28 $\stackrel{0;1}{\longrightarrow}$ 30) ;
  - au bout du délai d'ouverture rapide, la cellule D1 commande la fermeture automatique de son disjoncteur à la date $4400\,Ut$ (... $\stackrel{1400;1}{\longrightarrow}$ ... $\stackrel{1400;1}{\longrightarrow}$ ... $\stackrel{1600;1}{\longrightarrow}$ 41) ; au bout du temps de pré-confirmation du capteur, un défaut est de nouveau signalé à la cellule (41 $\stackrel{1000;1}{\longrightarrow}$ 52) ;
  - à la date $5800\,Ut$ (i.e. la fin du délai d'ouverture du cycle rapide de la cellule D2), la cellule D2 envoie la commande de fermeture automatique à son disjoncteur ($\stackrel{5800;1}{\longrightarrow}$ 56). Le défaut réapparaît et est signalé à la cellule D2 au bout de $\mathbf{T}_{PC}$, l'ouverture est alors immédiatement commandée (56 $\stackrel{1000;1}{\longrightarrow}$ 67 $\stackrel{0;1}{\longrightarrow}$ 68).

On visualise donc que l'on a un défaut multiple à la date $5800\,Ut$.

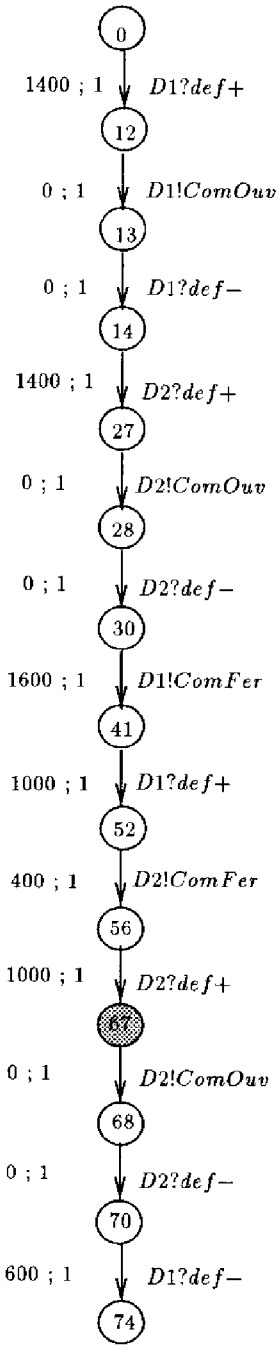

Figure 12: vue abstraite aux interfaces cellules/procédés ($T_{\mu C} = 1000\,Ut$)

Remarque : ici, compte tenu de la projection quantitative faite uniquement sur les états à l'interface cellule-capteur, nous ne pouvons pas évaluer la durée du défaut multiple. Il aurait fallu intégrer des projections sur l'état des perturbations électriques.

4.2.2.2 Cas 2

La vue abstraite est représentée sur la figure 13. Les analyses effectuées ici montrent les différences par rapport au cas 1.

- Analyse qualitative

  - les séquences des échanges entre chaque cellule et son environnement respectif sont identiques à celles du cas 1 jusqu'à l'état 62 où le défaut est de nouveau signalé à D1 ($... \xrightarrow{D1?def+} 62$) ;
  - puis, la disparition du défaut sur la ligne D1 est signalé en cours de fermeture automatique ($\xrightarrow{D1?def-} 86$) ;
  - enfin, la commande de fermeture est envoyée par la cellule D2 à son disjoncteur ($\xrightarrow{D2!ComFer} 93$) ;
  - à partir de l'état 86, la cellule D2 envoie la commande de fermeture à son disjoncteur ($\xrightarrow{D2!ComFer} 93$) pour effectuer la fermeture automatique. Enfin, le capteur informe la cellule D2 de l'état sans défaut de sa ligne ($\xrightarrow{D2?def-} 106$).

  On n'a donc plus d'existence de défaut multiple.

- Analyse quantitative

  - à la date 5400 Ut, le système (dans l'état 62) a effectué la même séquence que dans le cas 1 à la même date ;
  - à partir de l'état 62, la disparition du défaut sur D1 est signalée au bout de 2000 Ut à la cellule D1 ($62 \xrightarrow{2000;1} 86$), à la date 7400 Ut (i.e la disparition du processus de génération de la perturbation électrique) ;
  - à partir de l'état 86, la cellule D2 envoie la commande de fermeture à son disjoncteur au bout de 200 Ut ($86 \xrightarrow{200;1} 93$), c'est-à-dire à la date 7600 Ut. Puis, le capteur de la cellule D2 informe cette dernière de l'absence de défaut sur sa ligne au bout de $\mathbf{T}_{PC}$ ($93 \xrightarrow{1000;1} 106$).

  On vérifie ici, par rapport à la séquence obtenue dans le cas 1, que la fermeture du disjoncteur D2 est retardée de 2200 Ut ; ainsi, le défaut multiple est évité.

L'analyse de ce cas montre que le défaut multiple, présent au cas 1, est évité grâce à la prolongation de la phase d'ouverture rapide de par la connaissance *en temps réel* de la présence d'un défaut sur D1. La propriété d'évitement de défaut multiple est garantie car le retard entre l'apparition du défaut sur D1 et sa transmission sur le réseau à destination de D2 sont minimisés. Ceci permet de valider la contrainte imposée par l'équation 3.1 proposée au chapitre 3.

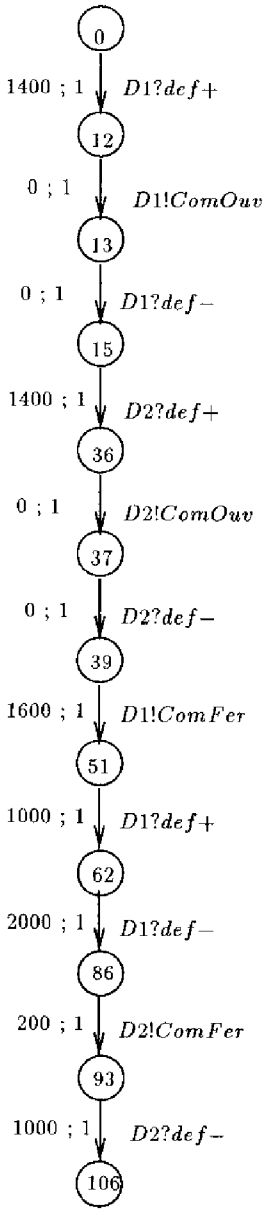

Figure 13: vue abstraite aux interfaces cellules/procédés ($T_{\mu C} = 500\,Ut$)

On en conclut que la période de mise à jour choisie dans ce cas permet de garantir les contraintes temporelles fortes des échanges temps réel.

5 Conclusion

Deux points de l'étude effectuée dans ce chapitre nous paraissent devoir être plus particulièrement soulignés :
- la structuration des cellules départ et de l'environnement de l'application de surveillance de poste de transformation d'énergie électrique HTB/HTA ;
- l'analyse sur l'évitement des défauts multiples dans l'hypothèse de perturbations électriques polyphasées sur deux lignes départ.

La structuration proposée a pour avantage de faciliter d'une part, la compréhension du rôle de l'environnement et de la fonction de la cellule départ et d'autre part, la représentation formelle avec les Réseaux de Petri Temporisés Stochastiques.
Ainsi, dans le module environnement, on distingue trois modules qui représentent : les perturbations électriques, l'intelligence du capteur (qui analyse le signal pour détecter les perturbations électriques) et le disjoncteur (que l'on ouvre ou que l'on ferme suite à des ordres de la cellule). De même, le module cellule départ comprend trois modules internes qui représentent : l'état de la ligne locale, l'état de l'autre ligne et les mécanismes fondamentaux de la sélectivité logique.

L'analyse de l'application (basée sur le principe de la sélectivité logique) a montré l'influence, sur la fonctionnalité d'évitement de défaut multiple, de la période de mise à jour sur le réseau FIP.

En effet, au delà d'une période limite de mise à jour, déterminée par l'application de la contrainte de Shannon et à partir des contraintes temporelles identifiées sur les messages entre les cellules réparties, les résultats montrent que les fonctionnalités de l'application ne sont pas remplies ; ceci amène le système dans des états de dysfonctionnements dangereux pour la sécurité des biens et des personnes. Par contre, lorsque les échanges sont réalisés sur le réseau FIP en dessous de la période limite, les fonctionnalités de l'application sont garanties.

Cette analyse nous amène à conclure que, dans le cadre de la mise en œuvre d'un système réparti temps réel sur un réseau local de communication guidé par le temps, le choix de la périodicité des mécanismes de diffusion des données à contraintes temporelles fortes doit être effectué de façon à minimiser au maximum les retards entre les échanges d'informations afin, de garantir au mieux les fonctionnalités des applications réactives réparties. En particulier, ceci est primordial dans le cas des systèmes continus de production d'énergie, qui est le cas que nous avons analysé.

Conclusion

Le travail présenté dans ce mémoire concerne fondamentalement la spécification, la modélisation et la vérification des fonctionnalités de commande et de surveillance d'une application temps réel répartie de EDF sur le réseau de terrain FIP.

Ce travail a été effectué suivant deux grands axes :

- développer une méthodologie générale, basée sur le cadre formel offert par le modèle "Réseaux de Petri Temporisés Stochastiques" (RdPTS), pour la spécification et l'analyse comportementale de systèmes répartis temps réel ; l'attribut général signifiant que cette méthodologie s'applique à n'importe quel niveau du système considéré, que ce soit dans la partie communication ou dans la partie contrôle de l'application ;
- appliquer cette méthodologie à une application réelle de grande complexité (système de surveillance de sûreté de fonctionnement d'un poste de transformation d'énergie électrique d'EDF réparti sur le réseau de communication FIP) et démontrer l'intérêt du modèle RdPTS pour spécifier formellement des comportements temps réel, vérifier des propriétés du système et évaluer des performances (fonctionnelles et de sûreté de fonctionnement).

Nous présentons maintenant les principales contributions et nous donnons ensuite quelques perspectives.

Contributions

- En ce qui concerne la méthodologie.
  - L'importance et la complexité des systèmes considérés est à l'origine du développement de la méthodologie de structuration et de modélisation de ces systèmes, afin d'en faire l'analyse et la vérification au moyen des Réseaux de Petri Temporisés Stochastiques. Les concepts suffisamment généraux de cette méthodologie la rendent appliquable à l'étude d'autres systèmes complexes qui possèdent des caractéristiques temporelles.
  - La méthodologie repose sur les éléments de la structuration hiérarchique et modulaire qui, à partir d'une vue systémique, permet d'obtenir une architecture détaillée en termes de blocs fonctionnels de base et d'interactions (locales et distantes) ; cette structuration s'applique à la fois aux fonctionnalités de surveillance et de commande d'un système de contrôle-commande et aux mécanismes de communi-

cation des réseaux de terrain.

Les contributions importantes sont :

* la définition d'une méthode de constructionde modèles par places dupliquées, basée sur le test à zéro et l'utilisation d'arcs inhibiteurs ; ce qui évite la désensibilisation et la réinitialisation des transitions temporisées sensibilisées par des places accédées en lecture ;
* la proposition d'une stratégie de construction : par fusion de transitions immédiates (et simultanément sensibilisées) et de places dupliquées pour la construction de modèles à partir des modules élémentaires d'un même sous-système (ceci préserve le synchronisme des évolutions internes d'un sous-système) ; par places de communication pour la composition de sous-systèmes adjacents (ceci préserve leur asynchronisme) ;
* la définition de modèles d'assertions logiques simples (telles que *et, ou, ou exclusif*) et complexes (telles que *tout est vrai, tout est faux, il existe, tout n'est pas vrai, tout n'est pas faux, il n'existe pas...*). Les modèles proposés facilitent le passage des descriptions de comportements, exprimés en langage naturel (spécifications fonctionnelles) et issus du cahier des charges et/ou de normes, à une modélisation formelle en RdPTS en vue de la validation des spécifications.

– Le modèle "Réseaux de Petri Temporisés Stochastiques", sur lequel a été basé l'ensemble des études, a permis d'appréhender naturellement et simplement la modélisation des systèmes à un niveau de détails élevés et pertinents, grâce à la modularité et la composition des modèles de modules élémentaires. De plus, ce modèle a permis d'une part, d'effectuer tout un ensemble d'analyses et d'autre part, d'obtenir différents points de vue à partir de vues abstraites.

• En ce qui concerne le réseau de terrain FIP.

La structuration proposée permet une représentation fine, précise et exhaustive des entités application et liaison à la fois en termes de modules internes et d'échanges internes (notions non spécifiées dans la norme et que nous avons définies) et de relations entre les échanges internes et les échanges externes (services spécifiés dans la norme). Nous avons proposé une structuration qui comprend :

– pour l'entité application producteur, les modules : écriture locale, rafraîchissement, indication de réception de variable de synchronisation et indication d'émission de variable ;

– pour l'entité application consommateur, les modules : lecture locale, promptitude, indication de réception de variable de synchronisation et indication de réception de variable.

L'analyse des mécanismes de validité temporelle (rafraîchissement et promptitude) associés aux services périodiques synchrones a été effectuée tout d'abord, en supposant

qu'il n'y a pas de pertes d'échanges et ensuite, en considérant deux hypothèses de pertes possibles (une perte d'écriture par le processus utilisateur producteur et une perte de trames de synchronisation sur le médium pour l'entité consommateur). Les résultats mis en évidence sont :

- le bon fonctionnement cyclique en l'absence de pertes qui est observé par les status de validité temporelle rafraîchissement et promptitude toujours vrais dans le service de confirmation de lecture destinés au processus utilisateur consommateur ;
- la capacité de FIP à détecter les conséquences de pertes à partir des status de rafraîchissement et de promptitude :
  - suite à une perte d'écriture, la confirmation de lecture retourne le status de rafraîchissement faux ;
  - suite à une perte de synchronisation, la confirmation de lecture retourne le status de promptitude faux ;
  - suite à des pertes consécutives d'écriture et de synchronisation, la confirmation de lecture retourne les deux status, rafraîchissement et promptitude, faux...

L'analyse quantitative des status de rafraîchissement et de promptitude, contenus dans la confirmation de lecture pour les processus utilisateurs consommateurs, a permis l'évaluation du temps moyen d'occurrence d'une erreur (MTFF) dans les différents cas d'hypothèses de pertes.

Cette étude, à la fois qualitative et quantitative, sur les mécanismes de rafraîchissement et de promptitude, nous a permis d'obtenir des résultats complémentaires sur : la vérification de propriétés à partir de ces mécanismes et l'évaluation de leur sûreté de fonctionnement ce qui constitue, à notre avis, une approche originale.

- En ce qui concerne l'application du contrôle-commande de poste HTB/HTA de transformation d'énergie électrique répartie sur le réseau FIP.

  La structuration proposée de l'environnement et des cellules départ a facilité, de par la modularité, la compréhension des comportements et des mécanismes des différents éléments. Ceci nous a amené à définir une décomposition de l'environnement comprenant : un module générant les perturbations électriques, un module effectuant le traitement et un module représentant le disjoncteur. Les cellules départ de l'application ont été décomposées de la même façon en : un module représentant l'image de la ligne, un module représentant la connaissance de l'état des autres cellules et un module réalisant les automatismes de la sélectivité logique, en cas de défaillances électriques.

  L'analyse de l'application de poste (basée sur le principe de la sélectivité logique) a montré que la période de mise à jour des données sur le réseau FIP a une grande influence sur les fonctionnalités d'évitement de défaut multiple : au dessus de la période limite de mise à jour (déterminée par l'application de la contrainte de Shannon à partir des contraintes temporelles identifiées sur les messages entre les cellules réparties), les résultats montrent que les fonctionnalités de l'application ne sont pas remplies. Ceci amène le système dans des états de dysfonctionnements dangereux pour la sécurité

des biens et des personnes. Par contre, lorsque les échanges sont réalisés sur le réseau FIP en dessous de la période limite, les fonctionnalités de l'application répartie sont garanties.

Cette analyse a permis de conclure que le choix de la périodicité des mécanismes de diffusion des données à contraintes temporelles fortes, dans le cadre de la mise en œuvre d'un système réparti temps réel sur un réseau local de communication guidé par le temps, doit être effectué de façon à minimiser au maximum les retards entre les échanges d'informations afin, de garantir au mieux les fonctionnalités des applications.

Enfin, nous tenons à signaler également que les spécifications complètes, en Réseaux de Petri Temporisés Stochastiques, de l'application répartie de contrôle-commande de poste, obtenues en suivant la méthodologie, ont servi à d'autres applications. En particulier, elles ont été traduites en SIGNAL, puis compilées en code C exécutable et implémentées sur une maquette d'essai du réseau FIP. Les résultats de cette expérience, décrits dans [SB96], montrent la qualité (correcte dès la première exécution) du système obtenu et la rapidité du cycle de développement.

Perspectives

Les travaux présentés dans ce mémoire peuvent être poursuivis en considérant les orientations suivantes.

- En ce qui concerne le réseau de terrain FIP, il est immédiatement envisageable d'analyser et d'évaluer la qualité des services fournis au processus utilisateurs : d'une part, en introduisant de nouvelles hypothèses de pertes, soit de trames de synchronisation pour le producteur, soit de trames de données pour les sites consommateurs... ; d'autre part, en considérant d'autres services et status tels que périodiques resynchronisés, ponctuels...

- En ce qui concerne l'application répartie de contrôle-commande de poste de transformation d'énergie électrique, les études suivantes pourraient être effectuées :

  - la vérification d'autres propriétés spécifiques telle que :
    - la relance de la cellule arrivée (prolongation de son délai de confirmation de défaut en cas de défauts multiples sur les lignes départ) ;
    - les commandes de traitement distantes entre cellules réparties tel que le déclenchement, par l'arrivée, d'un départ en cas de défaillances du matériel (capteurs ou disjoncteur) ;
  - l'évaluation du taux d'ouverture intempestive du disjoncteur arrivée lorsque des défauts multiples départ cœxistent en considérant les modèles complets des différentes cellules d'une topologie classique de poste (à condition d'avoir les moyens informatiques suffisants). En effet, le taux actuel d'ouverture de l'arrivée (de l'ordre de 10%) étant élevé, il serait nécessaire de quantifier précisément l'apport des nouveaux principes de la sélectivité logique basé sur l'utilisation d'un réseau de communication temps réel.

- proposer des extensions aux automatismes qui pourraient utiliser les status de validité temporelles des données à contraintes temporelles fortes contenus dans les services de confirmation de lecture afin, soit de ne pas considérer ces données périmées, soit de les estimer en utilisant des estimateurs d'états.

- En ce qui concerne la méthodologie de modélisation modulaire et de composition, il serait intéressant de l'utiliser pour l'étude de nombreux autres systèmes (par exemple, la norme internationale du Fieldbus ISA/IEC en cours de définition) dans l'objectif d'y apporter des modifications et des extensions d'une part, en terme de structuration des systèmes et d'autre part, en terme de représentation de structures génériques de comportements.

Bibliographie

[ACG92] S. Amar, E. Craye, and J.C. Gentina. Une méthode hiérarchique de spécification et de prototypage des systèmes de production flexible. *RAIRO-APII*, pages 483–514, 1992.

[AFN] AFNOR. Norme réenclencheur. Technical Report NFC 64-100, afnor.

[AFN90a] AFNOR. FIP. Architecture et Présentation générale. NF C46 601, 1990.

[AFN90b] AFNOR. FIP. Couche Application. Services périodiques et apériodiques. NF C46 602, AFNOR, 1990.

[AJ95] Y. Atamna and G. Juanole. Methodology for obtaining abstract views of state graphs labeled with probabilities and times. An example of application to a communication protocol. In *MASCOTS'95, Durham, USA*. IEEE-CS Press, Januar 1995.

[Ata93] Y. Atamna. RPTS, a tool for Stochastic Timed Petri Nets. In *Fifth International Workshop on Petri Nets and Performance Models, Tools Exhibitions*. PNPM93, Toulouse, France, October 1993.

[Ata94] Y. Atamna. *Réseaux de Petri Temporisés Stochastiques Classiques et Bien Formés : Définition, Analyse et Application aux Systèmes Distribués Temps Réel*. Thèse de doctorat, LAAS-CNRS, Université Paul Sabatier, Toulouse III, 1994.

[Bai89] S. Bailin. An Object-Oriented Requirements Specification Method. *Comm. of the ACM*, pages 608–623, May 1989.

[BB87] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In *Computer Networks and ISDN Systems*, volume 14. North Holland, 1987.

[BB91] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE, Another Look at real-Time programming*, pages 1270–1282, September 1991.

[BDT92] D. Boudebous, J.C. Derniame, and J. Tankoano. Une démarche méthodologique de conception de systèmes répartis. In *Int. Conf. on Software Engineering and its Applications*. Le Génie Logiciel et ses Applications, Toulouse, France, December 1992.

[BG91] J.P. Bourey and J.C. Gentina. Conception structurée et modulaire d'architectures de contrôle réparti en production flexible manufacturière. *RAIRO-APII*, pages 349-376, 1991.

[BG92] G. Berry and G. Gonthier. The Esterel Synchronous Programming Language: Desgn, Semantics, Implementation. *Science of Computer Programming*, pages 87-152, 1992.

[BJ88] G. Beuchot and R. Josserand. *Les réseaux locaux industriels*. Hermès, 1988.

[BJ94] N. Bergé and G. Juanole. Rafraîchissement et Promptitude dans le Réseau FIP : Modélisation et Evaluation au moyen des Réseaux de Petri Temporisés Stochastiques. In *Real Time System Conférence, Janvier 94, Paris, France.* RTS'94, Janvier 1994.

[BJS95a] N. Bergé, R. Jacinto, and M. Samaan. Design of a Transformer Station Control System distributed on the Fielbus FIP. In *Symp. on Control of Power Plants and Power Systems (SIPOWER'95).* IFAC, December (Rapport LAAS-CNRS 95269) 1995.

[BJS95b] N. Bergé, G. Juanole, and M. Samaan. Using Stochastic Timed Petri Nets for Modeling and Analysing an Industrial Application based on FIP Fieldbus. In *Invited Paper on Int. conf. on Emerging Technologies and Factory Automation (EFTA'95).* INRIA/IEEE, Paris, October 1995.

[BJSA94] N. Bergé, G. Juanole, M. Samaan, and Y. Atamna. Methodology for LAN Modeling and Analysis Using Petri Nets Based Models. In *MASCOTS'94, Durham, USA.* IEEE-CS Press, Januar 1994.

[Boo86] G. Booch. Object-Oriented Development. *IEEE Trans. on Soft. Eng.*, pages 211–221, February 1986.

[BPMM95] G. Bochmann, S. Poirier, and P. Mondain-Monval. Object-oriented Design for Distributed systems: the OSI Directory Example. *Computer Networks and ISDN Systems*, pages 571–590, Januar 1995.

[Bra83] G.W. Brams. *Reséaux de Petri.* Ed. Masson, Paris, 1983.

[BRV80] G. Berthelot, G. Roucairol, and R. Valk. Reduction of nets and parallel programs. In *Proc. of the advanced course on general Net Theory of Processes and Systems*, volume 84. LNCS, 1980.

[BS95] N. Bergé and M. Samaan. Poste de transformation d'énergie électrique : description fonctionnelle et formalisation au moyen des Réseaux de Petri Temporisés Stochastiques. Technical Report HR-64/95/001, EDF-DER, Janvier 1995.

[Cal90] J.P. Calvez. *Spécification et conception des systèmes : une méthodologie.* Masson, Paris, 1990.

[Car94] R. Carmo. *Le réseau DQDB : spécification, modélisation et évaluation de performances de mécanismes pour des services temps réel.* Thèse de doctorat, LAAS-CNRS, Université Paul Sabatier, Toulouse III, 1994.

[CFT86] J.P. Calvez, E. Friot, and Y. Thomas. Some microsystems design methodology and its application to industrial problems. In *IECON.* IEEE, 1986.

[Com91] M. Combacau. *Commande et surveillance des systèmes à évènements discrets complexes : application aux ateliers flexibles.* Thèse de doctorat, LAAS du CNRS, Université Paul Sabatier, Toulouse III, December, 1991.

[Cru91] D. Cruette. *Méthodologie de conception des systèmes complexes à évènements discrets : application à la conception et à la validation hiérarchisée de la commande de cellules flexibles de production dans l'industrie manufacturière.* Thèse de doctorat, Université de Lille, Février, 1991.

[Deu90] Deutsches Institüt für Normung. Profibus. norme 19 245, part 1 and 2, 1990.

[Ell88] J.P Elloy. Le temps réel. *TSI - rapport du groupe de travail du CNRS sur le temps réel*, 7(5), 1988.

[FP94] A. Finkel and L. Petrucci. Composition/décomposition de réseaux de petri et de leurs graphes de couverture. In RAIRO, editor, *Informatique théorique et applications*, volume 28. DUNOD, 1994.

[Fre92] L.B. Fredrikson. A CAN kingdom. Technical report, KVASER edition, 1992.

[Gen86] H.J. Genrich. Petri Nets: Central Models and their Properties, Part I, Predicate/Transition Nets. *LNCS*, 254:26–59, 1986.

[Hoa78] C.A.R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21, 1978.

[IEC] IEC, Norme IEC 65C. *Fieldbus Specification.*

[IEE92] IEEE. Carrier sense multiple access with collision detection; access method and physical layer specification. Technical report, ANSI/IEEE, 1992. (ISO/IEC 8802-3).

[ISA] ISA, SC 65C 105 et 106. *Industrial Automation Systems - Systems Integration and Communications.*

[ISO] ISO/TC97/SC21/WG1 Int. Standard, September 89, ESTELLE, a FDT based on an extended State Transition model. *Int. Standard, September 89.*

[ISO93] ISO. Road vehicles – Interchange of digital information – Controller Area Network (CAN) for high-speed communication. Norme 11898, ISO, 1993.

[JA91] G. Juanole and Y. Atamna. Dealing with arbitrary time distributions with the Stochastic Timed Petri Net model - Application to queueing systems. In *4 th International Worskop on Petri Nets and Performance Models, Melbourne AUSTRALIA*. PNPM91 in IEEE-CS Press, December 1991.

[JA93] G. Juanole and Y. Atamna. *Petri Nets based models and Communication Protocols*, volume 18, chapter Manufacturing Research and Technology : Modern Tools for Manufacturing Systems. Elsevier Publisher by Zurawski, R. and Dillon, T. S. Editors, 1993.

[JABF92] G. Juanole, Y. Atamna, N. Bergé, and J.M. Farines. Modelling time critical communication networks with Stochastic Timed Petri Nets. In *International Workshop on Real-Time Programming, Bruges BELGIUM*. IFAC/IFIP, June 1992.

[Jen84] K. Jensen. Petri Nets: Central Models and their Properties, Part I Colored Petri Net. *LNCS*, 254:248–299, 1984.

[KDK+89] H. Kopetz, A. Damn, C. Koza, M. Mulazzani, W. Schalb, C. Senft, and R. Zainlinger. Distributed Fault-Tolerant Real-Time Systems: the MARS Approach. In *MICRO'89*. IEEE, 1989.

[Kop] H. Kopetz. Six difficult problems in the design of responsive systems. In *Responsive Computer Systems*, Dependable Computing and Fault-Tolerant Systems. Springer Verlag.

[LL73] C.L. Liu and J.W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environnement. *Association for Computing Machinery*, 20(1):46–61, 1973.

[Llo90] J.Ch. Lloret. *Réseaux Prédicat/Transition étiquetés pour la modélisation et la vérification des systèmes informatiques répartis.* Thèse de doctorat, Université Paul Sabatier, Toulouse, 1990.

[LMT94] P. Lorenz, Z. Mammeri, and J.P. Thomesse. A state machine for temporal qualification of time critical communication. *IEEE south Eastern Symp. on System Theory*, pages 654-658, 1994.

[Lor94] P. Lorenz. *Le temps dans les architectures de communication : application au réseau de terrain FIP.* Thèse de doctorat, Institut National Polytechnique de Lorraine, CRIN, Nancy, 1994.

[Mer76] P.M. Merlin. A methodology for design and implementation of protocols. *IEEE Trans. on Comm.*, 24(6), 1976.

[Mil80] R. Milner. A Calculus of Communicating Systems. In *Lecture Notes in Computer Science*, volume 92. Springer-Verlag, Berlin Heidelberg, 1980.

[Mol81] M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models.* Phd thesis, University of California, Los Angeles, 1981.

[Nat80] S. Natkin. *Les réseaux de Petri Stochastiques.* Thèse de doctorat, CNAM, Paris, 1980.

[Ram74] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri Nets.* Research report, MAC-TR 120, MIT, February, 1974.

[Rei85] W. Reisig. *Petri Nets, an introduction.* Springer verlag, 1985.

[RN93] P. Raja and G. Noubir. Static and Dynamic Polling Mechanisms for Fieldbus Networks. *Operating Systems Review*, 27(3):34-45, july 1993.

[Rol95] P. Rolin. *Réseaux locaux. Normes et protocoles.* 5ème édition, Hermès, 1995.

[Sab93a] J.P. Sabathé. PCCN - la sélectivité logique des protections : clauses fonctionnelles. Technical Report HM-34/93/019, EDF-DER, 1993.

[Sab93b] J.P. Sabathé. PCCN - la sélectivité logique des protections : spécification formelle. Technical Report HM-34/93/020, EDF-DER, 1993.

[SB96] M. Samaan and F. Borgards. Application of a Synchronous Programming Approach: Developpment of An Experimental Distributed Transformer Station Control System. In *Real Time System Conférence, Janvier 96, Paris, France.* RTS'96, Janvier 1996.

[SG93] M. Samaan and B. Gaucheler. Evaluation du modèle de programmation des automatismes de poste au moyen du langage synchrone SIGNAL. Technical Report HM-64/0006, EDF-DER, Décembre 93.

[She94] A.W. Sheer. *CIM Computer Integrated Manufacturing.* Springer-Verlag, 1994.

[SM90] Y. Souissi and G. Memmi. Composition nets via a communication medium. In *Advances in Petri Nets*, volume 483. LNCS, 1990.

[Sol] G. Solignac. *Guide de l'ingénierie électrique des réseaux internes d'usine.* ELECTRA. Lavoisier TEC & DOC, Paris.

[Sou93] Y. Souissi. Composition de réseaux de petri et validation modulaire de systèmes distribués. In Hermès, editor, *Réseaux et informatique répartie*, volume 3. Hermès, 1993.

[Tho93] J.P. Thomesse. Le réseau FIP. *Réseaux et Informatique Répartie*, 3(3):287-321, 1993.

[TN89] J.P. Thomesse and P. Noury. *Communication models. Cient-server vs Producer-Distributor-Consumer.* Contribution ISO TC/184/SC5/WG2-TCCA, 1989.

[Y. 69] Y. Sévely. *Systèmes et asservissements linéaires échantillonnés.* Dunod Université, 1969.

[Zim80] H. Zimmerman. OSI reference model - The ISO model of architecture for Opens Systems Interconnexion. *IEEE Trans. On Comm.*, 28(4):425–432, 1980.

Thèse de Mademoiselle Nathalie BERGÉ

Titre
Modélisation au moyen des Réseaux de Petri Temporisés Stochastiques d'une application de contrôle-commande de poste de transformation d'énergie électrique répartie sur le réseau de terrain FIP

RÉSUMÉ
Ce mémoire présente des travaux concernant la spécification, la modélisation et l'analyse de systèmes de contrôle-commande temps réel répartis sur un réseau de terrain. La problématique de modélisation de tels systèmes, qui doivent satisfaire des contraintes temporelles, réside dans la maîtrise de la taille et de la complexité des modèles. Pour cela, une méthodologie de modélisation est proposée ; elle repose sur les concepts de structuration, de modélisation et de validation modulaires au moyen du modèle formel Réseaux de Petri Temporisés Stochastiques.

La structuration consiste en une décomposition hiérarchisée en blocs fonctionnels élémentaires. Une structure est proposée pour des entités de communication d'un réseau local temps réel, ainsi que pour un système temps réel de commande et surveillance réparties.
La modélisation repose sur la construction et la composition de modèles de modules élémentaires. Les principes énoncés concernent la définition de règles de composition. Des modèles d'assertions logiques sont également proposés pour faciliter le passage des descriptions de comportements exprimées en langage naturel, à une modélisation en Réseaux de Petri Temporisés Stochastiques.
La validation repose sur des étapes de composition partielle qui utilisent la modularité pour effectuer des vérifications ascendantes du comportement.

Cette méthodologie est appliquée au réseau de terrain FIP, ainsi qu'au futur système de contrôle-commande de poste de transformation d'énergie électrique d'EDF. L'étude du réseau FIP porte plus particulièrement sur les services et mécanismes périodiques de couche application pour les échanges temps réel. Les analyses qualitatives et quantitatives effectuées sur l'application de EDF, répartie sur le réseau FIP portent sur la vérification de propriétés fonctionnelles et de contraintes temporelles du système.

Mots clés : systèmes distribués, temps réel, contrôle-commande, réseau local de communication, réseaux de Petri Temporisés Stochastiques, modélisation, évaluation.

Title
Modeling with Stochastic Timed Petri Nets of a Power Transformer Station Control System Distributed on the Fieldbus FIP

ABSTRACT
This thesis deals with the specification, modeling and analysis of real time control-command systems distributed on a fieldbus. Modeling such systems which are characterized by hard real-time constraints is problematic due to the large size and complexity of the resulting models. To cope with this problem, a modeling methodology is proposed; it lies on the concepts of modular structuring, modeling and validation using the stochastic timed Petri nets formal model.

Structuring consists of a hierarchical decomposition of the system into elementary functional blocks. Specific structures are proposed for fieldbus communication entities and for real time distributed control systems.
Modeling lies on the construction and composition of elementary module models. Principles for module composition are defined. Logical assertion models are also proposed to facilitate the modeling, with Stochastic Timed Petri Nets of behavioral descriptions given in a textual language.
Validation is based on partial composition stages using modularity to allow ascending behavioral verifications.

This methodology is applied to the FIP fieldbus and to the future power transformer station control system of EDF. The study of FIP fieldbus concerns the analysis of application layer periodical services and mechanisms for real time exchanges. Moreover, qualitative and quantitative analyses are made on the EDF application distributed on FIP fieldbus for the verification of functional properties and temporal constraints.

Keywords: distributed systems, real time, control systems, local area network, stochastic timed Petri nets, modeling, evaluation.